Bachelor Thesis

A Monitoring system for a LoRa mesh network

Student

Alejandro Capella del Solar

Director

Felix Freitag

Tutor

Joan Sardà Ferrer

April 18, 2022

Information Technologies

Abstract

The internet of things(IoT) has been a pushing technology in the last years. As time goes by, the little devices have become more powerful and capable of doing more complex calculus. Among all these technologies one particularly has become quite mainstream in the field, these are the LoRa devices suitable to build low power wide area network (LPWAN). Phd. Roger Pueyo Centelles designed a protocol for LoRa mesh networks, and the Bachelor Sergi Miralles build a first approach of the protocol, afterwards Joan Miquel Solé made a stable version of it. Now we need to know more about the behaviour of this protocol over the field. In this Bachelor Thesis we created a whole monitoring system that will help us in the future doing analysis of a lot of experimental research with less effort
Keywords— IoT,LoRaWAN,mesh networks, monitoring

Dedication

To my beloved father, Jesús Capella Moner 1922 -2012 †
You live in me.

Acknowledgements

I want to thank deeply the time that my thesis director Professor Felix Freitag dedicated for the guidance of this work and for encouraging me to follow my criteria in the way of carrying out the statistical research that brought me finally to develop a whole monitoring system for LoRa mesh networks. Without your support and assitance that wouldn't have reached to a satisfying end.

I want to thank as well my comrade Joan Miquel Solé for improving the code quality of the given code that made my job a lot easier because in the very beginning I was absolutely unable to understand the code I was handling and I really feared of not reaching my academic goals.

I don't want to forget about Eloi Cruz Harillo who finished his TFG the previous January and helped me a lot to connect the boards via Wifi, when I was getting used to platformio and everything seemed so new and difficult for me and no of my attempts seemed to work the way they were supposed to .

I want to acknowledge as well the great job that Roger Pueyo did with the concept and design of the LoRaMesher, for placing in my way all these wonderful people that awake in me the passion of the research labour. I always will remember your role as a counsellor and guide to the world of LoRa communications which was an unknown world for me.

At last but not least I want to thank all the people that trusted in me in this 22 years journey that took me until this moment when I'm out to get my degree, despite all the times I failed or even got expelled from this faculty due to several health issues. Specially I want to thank computer science department Professor Luis Antonio Belanche who always encouraged me to finish my degree though I was about to reach 40 years old and still with a lot of subjects to be passed . I also have to mention my dear friend Alex Jurado Leyda, FIB engineer and statistics professor in the ETTAC, who always thought I was smart enough to get an Informatics engineering degree. I'm also thankful with all the people I met in

the way, in the FIB and outside it, friends, girlfriend, sisters and mother. I learned a lot from every of you guys.

Contents

List of Figures

List of Tables

Chapter 1

Context and Scope

1.1 Introduction

One of the most trending concepts in the IT world in the last years has been the internet of things (IoT). For this specific task, there is a pushing technology called LoRa in the field of Low Power Wide Area Networks(LPWAN).Colloquially speaking, an LPWAN is supposed to be to the IoT what WiFi was to consumer networking: offering radio coverage over a (very) large area by way of base stations and adapting transmission rates, transmission power,modulation, duty cycles, etc., such that end-devices incur a very low energy consumption due to their being connected[17]. LoRa is based on Long Range spread spectrum modulation derived from Chirp Spread Spectrum(CSS) technology.Most often it is used as part of the LoRaWAN architecture with a star-of-stars topology, but it can also be operated standalone, with a more flexible mesh network topology [18]. We have to be aware that when we are talking about LoRa communications, we may refer to two distinct layers, on one side we have the previous mentioned Chirp Spread Sprectrum technique, and on the other side a MAC layer protocol even though the LoRa communication system implies a specific network architecture.
The long-range and low-power nature of LoRa makes it an interesting candidate for smart sensing technology in civil infrastructures (such as health monitoring, smart metering, environment monitoring, etc.), as well as in industrial applications[17]. Since putting all the computing tasks on the cloud has been proved to be an efficient way for data processing, on the other hand the bandwidth of the network has come to a standstill, with the growing quantity of data generated at the edge, speed of data transportation is becoming the bottleneck for the cloud-based computing paradigm [25]. That's why when we are talking about the IoT world, we can not avoid going into the concept of edge computing. We mean by that allowing computation to be performed at the edge of the network in order to safe

Figure 1: Arduino uno rev3 original

bandwith and not having to go to the cloud to get the data we want. Here we define "edge" as any computing and network resources along the path between data sources and cloud data centers [25]. In order to satisfy this demmand of technologies with greater computing capabilities a whole family of new microcontrollers broke into market. From the very first single threaded Arduino in fig 1, a new shipment of programmable multi-threaded devices arrived. These were able of running machine learning algorithms, mesh network protocols or any kind of pervasive computing. They usually run code over a RTOS with a scheduler and they are able of distributing the load between the different cores.

1.2 Context

This Bachelor Thesis is the last step for the acomplishment of the needed credits to get my degree in Informatics Engineering with specialization on Information Technologies. Hence I will explain the story behind this work.

The UPC Phd student Roger Pueyo Centelles developed the theoretical basis of a minimalistic distance-vector routing protocol for LoRa mesh networks. To go a little bit deeper inside the details about this protocol I will say that it had a physical layer presenting multiple configurable parameters 1 :
The idea behind the pshysical layer that takes advantatge from previous multi-

Configurable parameter	Values
Radio Band	169, 433, 868, 915 MHz
Bandwith	62.5, 125, 250, 500 kHz
Transmission Power	14 dBm (EU), 27 dBm (USA)
Spreading Factor	6 to 12
Fec Rate	4/5, 4/6, 4/7, 4/8

Table 1: LoRa parameters,source :[18]

hop, mesh and routing for LoRa and LoRaWAN proposals is taking benefit from LoRa's Spread Factors' different range and orthogonality properties which allow for concurrent transmissions between different pairs of nodes, and introduces a novel multi-SF-aware ToA metric calculation that minimizes the total transmission time for a packet to reach the destination[18] Among all the features of this protocol we can focus in the following assets:

- Distance vector: best routes are calculated in a distributed way among nodes based on cumulative metric.
- Concurrent ,overlaid network, several "virtual" layered networks can operate on the same radio channel improving global throughput thanks to LoRa' SFs orthogonal properties
- ToA metric: based on the end-to-end packet transmission time, taking multi-SF capability into account.
- Layer 2 +3 hybrid: taking both layers as one makes easier to satisfy the low power requirements for embedded LoRa devices.
- Pro-active: the nodes of the network are periodically broadcasting new routes no matter how much data traffic are transmitting, refreshing routes and keeping them up to date readily available [18]
- Duty cicle-aware: this means that time on air limitations are enforced in the form of duty cicles, very important thing to have in mind when you are operating in ISM bands.
- Flexible and configurable : metrics, packet timing as well as other aspects can be fine tuned in order to fit specific use cases.

Taking this idea of LoRa mesh network protocol as a starting point, there has been done recently a FIB TFG which did a first implementation of it on c++ to be run on a board called TTGO T-Beam ESP32. This library though apparently

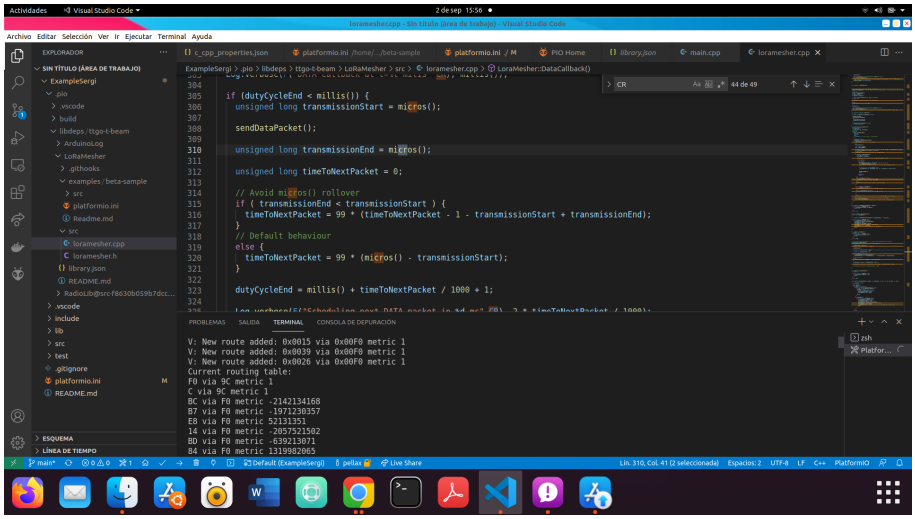


Figure 2: Example of negative values dropped out by the serial port, source :PlatformIO IDE

seemed to work, after some time of being deployed some negative values suddenly appeared for no reason by the serial port which means that something wrong is going on out there as we see on figure 21.
My first task on this Bachelor Thesis will be to find out what's wrong with this first implementation.Once I have fixed this, I will have to deploy a server to collect data from all the boards through a Wifi connection about all packets they transmit , maybe the content of the routing tables, still I have to think what metrics will meet better my goals. Next step will be to make a data pre processing and get some graphics and conclusions about. And repeat this experiments in different scenarios, distance, obstacles etc so we can figure out what is its performance .
So we will have the UPC as our main stakeholder because this study will be the first approach on field about this product 100% born in this house, as a collaboration between two faculties, the ETSETB and the FIB all gathered in the AC research department.

1.3 Justification

In IoT world, as well as in edge computing, is important having communication protocols that can function at Long Range maintining the resistance to multipath fading and to noisy environments.Thanks to LoRa systems have some configurable parameters and a Forward Error correction that makes them a good choice for noisy situations, for there is a trade off between speed transmission and communication range, being available the chance of demodulating the signal in the receivers, since

Scope	Parameter	Range	Default
Lora	Minimum SF	SF7, . . . SF_max	SF7
Routing	Maximum SF	SF_min, . . . SF12	SF12
Routing	Metric	ToA, HC, ETX, RSSI	ToA
Routing	AVG routes broadcast period(SF_min)	0 . . . ∞	60s
Routing	Routes expiry time (SF_min)	0 . . . ∞	300s
Routing	Max. routes to a node	0 . . . 2	2
Routing	Max. total routes	0 . . . 1024	1024
Routing	Routing/Data traffic prior.	n:1	10:1
Routing	Forward/Local traffic prior.	n:1	10:1
Regulation	Duty cycle (%)	0.1-100	100

Table 2: Multi hop distance-vector routing protocol options, source : [18]

they are using different and quasi orthogonal spread factors [18] Besides this there are some parameters , concretely the ones in table 2, that can be fine tuned to adapt it to different scenarios and we have to know extra info about more extreme conditions and see what happens and what can be improved.

1.3.1 Problem to solve

This brand new protocol has never been implemented properly nor tested in a real field work environment . We do not have any clue about its real performance and we don not know wether it needs to be fine tuned or modified to enhance its usability for the real world. For this reason we must make a first approach to its behaviour in a real use case and check the routing tables being formed if possible, and track all the packets throug the whole network to get some conclusions about. In more technical words we have the following explanation: since direct communication between a pair of nodes can occur at different SFs, they are indicated by the fastest (i.e., the lowest) SF possible. Communication between distant nodes that are not directly connected is made by multi-hop packet forwarding, using the routes calculated by the routing protocol. While most of the links in the diagram are symmetric, a few of them require different SFs in each direction to achieve a successful communication. This case can happen in scenarios with heterogeneous hardware or environmental conditions, either temporary or permanent[18]. For all this reasons a complete on-field work is needed.

1.3.2 Main Concepts

1.3.2.1 LoRa

LoRa is a long-range, low-power, low-bitrate, wireless telecommunications system, promoted as an infrastructure solution for the Internet of Things: end-devices use LoRa across a single wireless hop to communicate to gateway(s), connected to the Internet and which act as transparent bridges and relay messages between these end-devices and a central network server.[17]

1.3.2.2 IoT

Internet of Things is the Connections of embedded technologies that contained physical objects and is used to communicate and intellect or interact with the inner states or the external surroundings.Rather than people to people communication, IoT emphasis on machine to machine communication.[28] The Internet of Things (IoT), being a "network of networks", promises to allow billions of humans and machines to interact with each other. Owing to this rapid growth, the deployment of IoT-oriented networks based on mesh topologies is very attractive, thanks to their scalability and reliability (in the presence of failures)[19]

1.3.2.3 Mesh networks

In mesh topologies, network nodes are directly and dynamically connected in a non-hierarchical way, thus allowing many-to-many communications (among nodes cooperating with each other) to efficiently route data from a generic source to a generic destination[19]

1.3.2.4 Multihop

Multihop mechanisms propose intermediate nodes to forward messages from other end-devices. The intermediate nodes in each hop could be either an end-device or a gateway, performing simple relay functions or complex routing protocols, forming different mesh topologies[20].

1.3.2.5 LPWAN

Low Power Wide Area Networks were developed to provide a feasible solution for applications that require a wide area coverage and energy efficiency [20]

1.3.2.6 LoRAWAN

Long RangeWide Area Network (LoRaWAN) networks are one of the most studied and implemented LPWAN technologies, due to the facility to build private networks with an open standard. Typical LoRaWAN networks are single-hop in a star topology, composed of end-devices that transmit data directly to gateways. Recently, several studies proposed multihop LoRaWAN networks, thus forming wireless mesh network [20]

1.3.2.7 Data preprocessing

Refers to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the data mining process[23].

1.3.2.8 Data collection

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. Data–gathering methods are often loosely controlled, resulting in out–of–range values (e.g., Income: –100), impossible data combinations (e.g. , Sex: Male, Pregnant: Yes), and missing values, etc

1.3.2.9 Edge computing

Edge computing is a bridge for realizing the convergence between physical space and cyber space. Large numbers of physical objects produce a huge amount of data that needs to be efficiently processed in the edge side.Edge computing moves the capacities of computing, communication, storage, services and control to the system edge in close proximity to the end devices and users. So that it can provide smart edge services with low latency, high reliability high bandwidth efficient and high security[22]

1.3.2.10 Data cleansing

Data cleansing is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset

1.3.2.11 Data Mining

Data Mining is a process of extracting and discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems [27]

1.3.3 Implied Metrics

In order to get usable information from our experiments we will identify and select metrics which will help us to understand the protocol performance.

1.3.3.1 Time on Air(ToA)

The SF is a key element of the LoRa radio technology, as it poses a trade-off between the transmission reach and the time required to send a packet. Roughly, switching to a SF one step higher (e.g., SF7→SF8) doubles the transmission time (or halves the transmission speed, roughly), while increasing distance around ×1, 4. Throughput, or transmission time, also have a direct relation with the power required to transmit a packet, which is specially critical in battery powered devices. This is usually the case in the context of LPWAN for the IoT domain, where radio channel occupation and power (i.e., energy) are scarce resources[18]
According to this metric the protocol calculates the distance between nodes,thus the routing tables follow the same logic. We have to know not only how good is the performance of the routing tables in a real scenario, but ensure the durability of the battery if possible.

1.3.3.2 Routing tables

All the nodes in the mesh network are periodically receiving updates from neighbour nodes. Being a DV protocol, the table consists of a list of all the nodes known to be in the network, the neighbor through which to reach them, the path cost and the route expiry time. We have to know how trustful are these routing tables to know the real behaviour of the protocol. For this reason we will set up different scenarios to test how this routing tables are formed.

1.4 Project Scope

Currently the performance of LoRa mesh has only be simulated in the PhD thesis of Roger Pueyo. In my TFG we aim to experiment the performance of the LoRa mesh with real devices. In other words we will say
that this work will cover the validation of the LoRa mesh protocol implementation and its evaluation on field experiments. To do so we will create a mesh network out of 4 TTGO T-Beam esp32 boards and we will try try to have them transmitting packets for a couple of hours. We will try it in indoor environment and outdoor with different distances. We have the choice of taking off the antennas to simulate long distances and reduce the power and scope of the signal.

1.4.1 Methodology

This work is aiming to be a proof of quality of a new protocol devoloped by one of our students.

1.4.2 Risks

In section 2.8 there's a list about all the risks I might find along the way of this research and several alternative plans just in case one of this obstacles were unavoidable, nevertheless in this section we will do a little review about what obstacles we might have

- Since I will be working directly with the implementation of the protocol a deep knowledge about how this works is needed.
- Dealing with someone else code is always tough .
- I have no idea about how the RTOS the TTGO T-Beam boards works with.Task management may become an important issue since the hardware won't be only transmitting data packets between them, but sending periodically data through the wifi connection to the server. If I have technical issues I will have to ask for help. Some things like connecting the boards to the wifi are crucial. If I can not do it due to an error of the libraries, as I point out in coming sections, that would mean around a whole week of extra developing time.
- I have to develop a server. Maybe I can take an example from previous works running over these boards but maybe it doesn't work properly and I have to do it again on my own.
- Data cleansing, data collecting and data preprocessing in a real field work it's something really new for me, and it would be hard enough to deserve a whole Bachelor Thesis by their own. But in this case there's quite a lot of work to do before I reach that point.
- All the above reasons may delay our goals when we face meeting our deadlines.
- Personal issues: as I point on section 2.8 my health is delicate and the levels of stress I can handle is limited so I will have to be very strict with my rest and work timetable so I can unwind properly and get my mood balanced. To sum up we are in the middle of pandemics and we have to stay alert with our environment.

- Lack of time in my everyday routines. As I point out in the following sections, I'm in the middle of an internship, and I'm carrying out my last subject in the informatics degree, all that time is time I can not dedicate to the thesis.

For all these reasons I will try to follow these steps carefully

1. Get the boards connected to a Wifi network.
2. Develop a server in node.js to collect the data.
3. Containerize the server with docker to make the deploy at home or maybe in a FIB server.
4. Add the code to the boards to send the data I want to collect to the server.
5. Collect a few data for a sample and focus on the negative values to see what's wrong in the implementation.
6. Fix the implementation until there are no negative values anywhere.
7. First experiment with four boards exchanging data between them during a couple of hours.
8. Take off the antennas of the boards and repeat the experiment.
9. Some additional experiment with other alternative scenarios ?
10. Data cleansing and Data pre processing with Elastic Search using LogStash to load the data.
11. Data visualization with Elastic Search.
12. Write down conclusions.

    It's also essential having a realistic time plan and meeting the deadlines, for It's quite probable that unexpected issues may appear in the way and I can't afford any delay due to technical problems with the board, the server, the integration of both parts or the data collection process since the statistical study will also take its time.

1.4.3 Goals

The main goal of this project is to measure the determine how trustful this LoRaMesher protocol is. To acomplish this objective, we have a couple of sub goals that need to be satisfied :

- Get an implementation of the LoRaMesher protocol usable and free of errors and deploy it to a few boards.
- We must build a server to gather all the data from the logs coming from the boards.
- Determine the metrics that will help us to quantify the fairness of the protocol .
- Get a visual representation of the data most suitable for the conclusions that we would like to reach.

1.4.3.1 Functional requirements

Besides the main goal of the study, which is defined in the lines above, we must meet another requirements that we need for the proper working of the system to make the statistical analysis.

- The implementation of the LoRaMesher has to be free of errors, no negative values should appear as long as te boards will be turned on transmitting data .
- The implementation must be trustful and portable to any ESP32 like architecture boards.
- The RTOS of the board must balance the work load among all the cores at the time of transmitting the logs and sending the routing data packets
- The modifications that we will be making must follow the same idea of the code we will inherit in order to not make a code made out of patches.

1.4.3.2 Non-Functional requirements

Next will be talking about non-functional requirements, that is to say, those requirements we have to meet not being these strictly about the working of the whole system we will be making the study about, but we should consider from the very beginning to the correct development of the whole study we are aiming to carry out.

- The metrics from the statistical study must give us an idea of the performance of the LoRaMesher.
- The visualization of the data must be informative and useful as a tool of measurement of the protocol in different environments.
- The final code must allow future improvements and new functionalities to be added.

1.4.4 Main elements

1.4.4.1 Hardware

ESP32 We will use this board in figure 3 with LoRa cappabilities to make a deploy of the LoRaMesher library with the implementation of the multihop distance vector routing protocol to transmit data packets and collect it through a server.

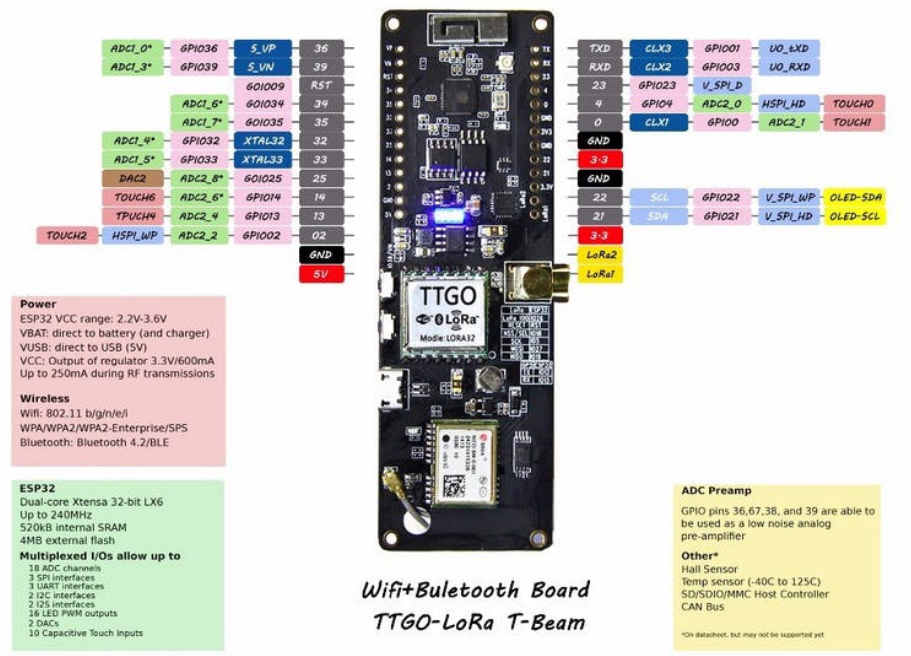


Figure 3: TTGO T-Beam ESP32 Datasheet,source: TTGO T-Beam documentation

1.4.4.2 Software

PlatformIO Professional collaborative platform for embedded development Features

- A lightweight but powerful cross-platform source code editor.
- Smart code completions based on variable types, function definitions, and library dependencies.
- Multi-projects workflow with easy navigation around project codebase, multiple panes, and themes support.
- Seamless integration with PlatformIO Home (UI) with board and library managers.
- Intuitive project wizard and a wide range of example projects.
- Built-in Terminal with PlatformIO Core (CLI) and powerful Serial Port Monitor.

Trello This is a collaboration tool that organizes your projects into boards. Though this thesis is an individual work, I want to keep a record of the thing to do and thing done to acomplish the objectives before the deadlines according to the time planning.

Docker Docker is an open platform for developing, shipping, and running applications. To make the deploy of our server I'll be using containers to avoid messing out my host configuration which is something very probable since I'll be working with linux.

Node.js It's a well known backend framework that I'll use to develop a simple server to collect all the data coming from the boards.

MongoDB It's a NoSQL database which most characteristic feature would be his flexibility and scalability. I'll be using it to keep all the data and after that export it to the following step.

ElasticSearch I'll be using ElasticSearch to all the statistical visualization, Inference and whatever It may be useful for.

Kibana It's an UI to interact with my ElasticSearch engine and get the proper visualizations of the data, and in general terms it gives access to the Elastic Stack in a very user-friendly way.

LogStash It's a Data processing pipeline from the server side. I'll use it to feed with data the ElasticSearch engine, it will help me with the data transformation and probably with the data cleansing.

Chapter 2

Time Planning

To carry out this Bachelor Thesis we have scheduled the number of hours that I have to dedicate daily to reach out the objectives in this work. We have forseen to spend 4 hours daily, from Monday to Friday to TFG tasks such as programming the board, developing the server or data collection. We will invest the time on the weekends to retrieve any chunk of time during the week that for any reason I was unable to work properly on the thesis because I have some other subjects that need to be attended besides my studies.
Apart from my thesis, I'm also enrolled on my last subject on the degree and that means that I will need some extra time to reach also the deliverables and commitments with the people on this subject, so a very strict time planning is needed in order not to get caught by the deadlines.

The project started the previous weeks, concretely September the 13th. Nevertheless, this first weeks of the current course I've been busy with the new subject and writing the first deliverable and it took quite a lot to acomplish. From now on I will follow strictly my timetable otherwise I won't be able to fulfill muy goals. The thesis will be delivered the Monday 17th of January of 2022, and the lecture will take place the week of the 24th of January.

We will spend around 728 hours developing this work.That makes a mean of 20 hours a weeks, and in order to not loose the rythm of work we will keep a log of hours worked. We have an advantatge that is that I've been working the summer previous on it so many of the tasks of the project management and previous work are already done, in the gantt diagram appeare like tasks done in parallel at the beginning of the project.

2.1 Task description

In this section we will enumerate and describe the tasks and subtasks needed to acomplish the goals on this TFG. I have built a table with the details to make it more understandable. In the following sections I will make a short comment about every planned task.

2.2 Project Management

First of all we should define the scope and objectives of the project according to the stakeholders needs, in this case the UPC. For this reason there was an initial meeting with my director of this thesis. Therefore there has been a continous communication channel between him and I to validate the scope of the project and ensure that we were aiming for the same goals.

PM.1 Scope of the project

Since we are being given a project from some other student, first of all we had to take a look to the code and see what parts were working and which ones were still undone. This task lasted about 15 hours

PM.2 Time planning

We have to be very careful with the timings because there can be several issues and delays during the whole process. If we don't invest enough time to every task, for sure we won't acomplish the goals. If this wasn't enough, in this document we will talk about obstacles and possible solutions.
In this case we had to do a time planning in a few hours due to other issues during my studies, it took about 10 hours.

PM.3 Budget

The main source of expenses will be on time of the researcher, my time. The equipment needed to carry out this enterprise will be my laptop, the four boards, the batteries and wires.The most difficult thing will be to calculate the power consumed by all these hardware. Lets say this will take a day of work or maybe less, about 4 hours.

PM.4 Sustainability Report

For this task a little bit of research will be needed to forecast the social, economic and enviromental impact for the planification, developing, data collection and statistical study tasks. In one day full time will be ready, 4 hours.

PM.5 Meetings

To ensure that the thesis is going the right way I will have weekly meetings with my director to receive guidelines and discuss the results and issues over the course of the thesis. Every meeting will be about 30 min. 14 hours in total.

PM.6 Documentation

I will be writing the reports and all the documentation as I get through all the steps along the research of this work. If we gather all the time I will spend writing docs with latex they will be about 50 hours of the final result of time.

PM.7 Presentation

I will have to get ready for the final lecture, prepare some powerpoints with the results and explanaitons of the process, It will take me the last 10 hours of the estimated total time.

2.3 Previous Work

When we decided to start over with this project, there were some things that needed to be done first.

PW.1 State of Art

Refering to the given code of the board from the previous TFG student I had to look what were the errors coming out from the serial port and what were the reasons behind this behaviour according to what was meant to be the protocol LoRaMesher. Understanding the protocol was also essential to acomplish this task. Its estimated duration is about 10 hours

PW.2 Setting up the work environment

Besides this previous task, there was also a time to get in touch with the developing environment, the platformio IDE, the setup, and the boards and to get the previous

code running in the same way as my predecessor had. Since there can be some issues due to different Operative Systems setups, we forecast a whole week of work to do this, 20 hours.

2.4 Developing

We have to develop several things in this project, we have among many other things we have to finish the protocol implementations properly, we have to transmit the logs to the server and we have to develop the server itself

D.1 Developing the Board

We have the following subtasks to reach out all the forecast tasks in this thesis. These are

1. Fix the protocol :80 hours
2. Get the boards connected to the wifi.This maybe is easy maybe is not: 80 hours
3. Transmit the logs to the server.We don't know wether It may appear synchronization issues or any other kind,lets estimate 100 hours of work

D.2 Developing the server

To develop the server we will use node.js and It's something that can be quickly , in a week of work(80 h) we could have a server done and usable.

2.4.1 Integrating parts

We have two main tasks here

1. Containerizing the server :40 hours
2. Test: we will collect a small chunk of data to see that we have a operative data channel between the boards and the server. I estimate a week of work :40 hours.

2.4.2 Data collection

If everything is working we will have a whole day doing data collection, maybe two.20 hours of work.

2.4.3 Data preprocessing

We have to clean up the data, select the data and in general terms, get the data ready to make the graphics. 60 hours.

2.4.4 Graphics

We will spend the last 60 hours doing graphics of the metrics from all the experiments and writing down conclusions but these last ones are included into documentation time .

2.5 Resources

2.6 Human resources

I'm afraid I'll be the only one working on this project, nevertheless I must say that the previous TFG student, Sergi Miralles, did already a good job with the first implementation of the LoRaMesher, as well as the Phd Student Roger Pueyo Centelles who left a good collection of bibliography talking about his protocol and its state of art . But from this point I'll be the one in charge of validating the reliability of the LoRaMesher. Even though being the only one working on it I'll describe the six roles I'll assume during this research

- Project Manager : the person in charge of the coordination and planification of the project and the one who writes all the documentation.
- Researcher: Designs the experiments, methodology and techniques in order to acomplish the goals of the thesis .Also contributes to the documentation.
- Developer: Programs the boards and the server .
- Dev ops : Helps to integrate the different parts of the product and improves the quality of the final result.
- Statistic: Deals with the data preprocessing, data cleansing and gets information out of the graphics and studies.
- Tester: Ensures the good working of every element of a system,software or Hardware.

2.7 Material Resources

Besides me, I'm going to enumerate what resources are available to carry on this TFG

- My Laptop : Acer Nitro Intel Core I7,used for developing the server, fixing the LoRaMesher, deal with the data and write down the documentation.
- 4 TT GO T-Beam boards used for deploying the implementation of the protocol, transmit data packets, sending data to the server, they are the main hardware element of this project.
- Wires and batteries to plug the boards into my computer and being flashed and having the chance of moving around the boards without being plugged in.
- platformio IDE: the choosen IDE to fix up the LoRaMesher implementation.
- Overleaf : to write down all the documentation.
- Gantter : to make gantt diagrams in an easy way on the cloud.
- Trello to keep a log of the things done and the things to do.
- Elastic Stack: essential tool to deal with data, data preprocessing and doing graphical work with data.

Here in table 3 we have a record of the tasks with the resources, the amount of hours dedicated to each task and the roles involved.

2.8 Risk Management

Several things can go wrong during the course of doing this TFG. In the following lines we will a take a look at them and we will try to figure out what could we do to avoid them and what would it be an alternative plan if there is any.

- Not meeting the deadlines. Some of the tasks are planned in a very optimistic way but whoever who has ever worked with this kind of hardware knows how difficult can be sometimes to get the things done the way it was planned. If there is something that is not working properly and I've been through the half of the forecasted time for that task the only way out would be asking for help. Luckily I keep a very good relationship with every professor that I've had in this college and besides my Bachelor Thesis director I think I would

Id	Task	Time	Dependences	Resource	Role
PM	Project Management	107h		Laptop, Overleaf,Trello	PM
PM.1	Scope of the project	15h		Laptop, Overleaf, Trello	PM
PM.2	Time planning	10h		Laptop, Overleaf, Gantter,Trello	PM
PM.3	Budget	4h	PM.2	Laptop, Overleaf,Trello	PM
PM.4	Sustainability Report	4h	PM.3	Laptop, Overleaf, Trello	PM
PM.5	Meetings	14h		Laptop	PM,D,T,S,DVOPS,R
PM.6	Documentation	50h		Laptop, Overleaf, Trello	PM,R
PM.7	Presentation	10h	DT.3	Laptop	PM
PW	Previous Work	30h	Previously done		R,D
PW.1	State of the Art	10h	Previously done	Laptop,Trello	R
PW.2	Setting up the work environment	20h	Previously done	Laptop, Platformio, Trello	D
D	Development	340h	PM.4	Laptop, Platformio	
D.1	Development of the board	260h		Laptop Platformio, TT Go T-beam board,wires, Trello	D
D.1.1	Fix the protocol	80h	I.2	Laptop Platformio, TT Go T-beam board,wires, Trello	D
D.1.2	Connection of the Wifi	80h	PW.2	Laptop Platformio, TT Go T-beam board,wires, Trello	D
D.1.3	Transmit the logs	100h	D.1.2	Laptop Platformio, TT Go T-beam board,wires,Trello	D
D.2	Development of the server	80h	D.1.3	Laptop, Trello	D
I	Integration	80h		Laptop Platformio, TT Go T-beam board,wires, Trello	
I.1	Containerizing the server	40h	D.2	Laptop, Trello	DEV OPS
I.2	Communication test	40h	I.1	Laptop Platformio, TT Go T-beam board,wires,Trello	DEV OPS ,T
DT	Data Treatment	140h			
DT.1	Data collection	20h	D.1.1	Laptop,TTGO T-Beam, board, batteries, Trello	S,R
DT.2	Data preprocessing	60h	DT.1	Laptop, Elastic Stack, Kibana, Trello	S
DT.3	Graphics	60h	DT.2	Laptop, Elastic Stack, Trello	S
	Total	697H			

Table 3: List of tasks, including dependences, resources and time. Roles: PM stands for Project Manager, D for developer, T for tester, S for statistic, DVOPS for DEV OPS, R for researcher, source:own elaboration

know exactly what should I ask to whom in case that there was something not going the way I planned. So all these people would be my additional resources.

- I'm through an Internship on a company. That means that I don't have all the time in the world to do this research.Probably my work there takes away from me my most productive hours of the day increasing my stress and damaging my well being.

- I have a 65% of disability due to mental health issues and I don't know wether I can handle out high levels of stress. I'm used to being stressed out due to studies, but maybe this may become unbearable which is something I don't know right now. Lets cross fingers and I hope that I will be ok . To prevent that I must be very strict with my sleeping time and find some time every day to lose and enjoy otherwise It could be dangerous for me.

- Pandemics: Just like any other one in the planet I'm in the middle of the covid 19 pandemic and though I'll get fully vaccinated the coming weekend, It's definetly something that I must keep in mind so I must beware of going into crowded places or big meetings. By now I would stay at home rather than going to social activities.

- Too ambitious goals: If I realize that the whole work is too much the approach of the work may be slightly different. We must be aware that it's not little work what I'm trying to do here and many things may not go the way I initially planned. If that happens I would ask to my Director what would be the best thing that I could do to have something valuable in order to get assessed. Maybe should I leave some experiments aside, maybe should I do a deeply research about the obstacles I found and what are the solutions I've been dismissing and why explaining deeply all the path I've been through.

2.9 Project deviations

This thesis was meant to be delivered at the end of January. Nevertheless due to all my must do tasks, the last remaining subject at the FIB, my job as trainee IT engineer and my limited abilities became in not being able to dedicate the amount of hours I had to acomplish my goals at the right time. I couldn't deliver results for the previous report and I suffered from several delays. I will make a brief description of the deviations of the initial project.

2.9.1 Time deviations

Since I was unable to meet the goals at the right time I had to rebuild a time planning for the coming months taking profit from the extra time I had by the end of January. Only having to attend to my job by the mornings, I had all the evenings to dedicate to the thesis. That's why I create a new Gantt diagram with extra tasks showed in appendix C and new time planning for the remaining jobs to get the desirable results.

2.9.2 Cost deviations

We are counting 20 hours a week of effective work. This will mean a certain costs besides the initial budget. Here are the details 4
This is an aproximation because we were unable to execute all the previous job hours, but it is a realistic record of all the hours we've been doing through all this extra time . We had a contingence plan of 3895€ but in order to know what would it be the final cost of the project we should make a readjustment of the whole task division, as we will explain in the coming sections, the main goals of this thesis has been modified briefly.

2.9.3 Initial conditions deviation

When I started with this thesis I was working with a very precarious version of LoRaMesher. I did what I could during that time but I did not meet the deadlines properly. Luckily, after christmas, Joan Miquel Solé came out with a new version and this was a very good new because his code was a lot more ordered, much more easy reading code, more easily extendable, organized and well thought. And this event was crucial for the right development of the thesis because I must admit that whit the previous version I was totally working blindfolded . Once I had the new code working I started being able to rethink what was about to do to get the data out of the boards in an efficient way .

2.9.4 Goals deviation

Once I established a communication channel between the boards and our node server I found myself trying to get the data out of the mongo container and I found it quite messy doing all the process by hand and I thought that there must be somehow of doing this automatically that would save a lot of time in the future for extracting plots out of the data without any human action in the middle. So I benefit from the Elastic Stack, I discovered the elastic javascript client and I thought that this would fit my needs perfectly. Doing it like this would be a lot easier than making a data cleansing, data transformation, loading the data through logstash every time we wanted to make an experiment and a lot easier and faster. So my thesis focus would change a little bit from getting results out of the experiments with the boards, to build a real-time monitoring system that will allow us to make experiments for longs periods of time, not only five minuts as we were supposed to do in the very first moment. However we will try to make some experiments as well and validate our monitoring system.

ID	Task	Work hours	Roles	COSTS	COSTS+SI
S.1	Studying new code	20	R,D	740€	962€
C.1	Coding the new task	20	D	440€	572€
CN.1	Connection with node server in containerized environment	20	DVOPS	400€	520€
CN.1.1	Connection test with node	14	DVOPS,T	490€	637€
RK	Research with ElasticSearch and Kibana	20	R	300€	390€
D.1	Dockerization Kibana, ElasticSearch	20	DVOPS	400€	520€
I.1	Integration node, mongo, ElasticSearch	20	DVOPS	400€	520€
VE.1	Validation experiments	20	R,S	700€	910€
DP.1	Deploying to a public Ip	42	DVOPS	840€	1092€
AD.1	Additional documentation	257	PM	6682€	8686€
	Total additional costs			11392 €	14809 €

Table 4: Table of extra expenses,source:own elaboration

Chapter 3

Economic Management and Sustainability

Once we have defined the risks and how can we avoid them and what would it be their alternative ways of reaching our research goals, we must estimate the costs needed for the correct development of the research.
We can identify several kinds of costs, coming out of personal , material, extra expenses,work spaces or additional hardware.
Additionally we will set up a contingence plan in order to face off the possible the obstacles and extra expenses. For that that reason we will plan a batch to assume the unexpected issues that may appear during the thesis.

3.1 Budget

3.1.1 Staff costs

We have defined 6 main roles for the development of this project. We have made an approach of the wage of every role taking info from the web page glassdoor.com [1] that has a record of the wages of all around the world. We have taken the means for the city of Barcelona for every role as we show in table 5
We have the details of the costs of the work hours per task on the table 6. Notice that we have multiplied by 1.3 to get the value from the hours of work, plus the costs of the social insurance.

3.1.2 Generic costs

From the time planning, we have divided the whole project into tasks considering for every task the human resources . The work will be developed from Monday to

Roles	Cost per hour
Project Manager	26€
Researcher	15€
Dev ops	20€
Developer	22€
Tester	16€
Statistic	20€

Table 5: Staff costs, source:glassdoor.com

Friday in my home, so we have considered that my 6m2 room for the computer as an office, the market price of a place like this in this neighbourhood costs about 25 € per m2 and we have multiplied by 5 which is the number of months that this project will last. In addition we have considered the internet connection during this five months, about 50€ per month.
We shall not forget the Software costs. We are using free and mostly open source software, nevertheless some of them do have some cost that we must consider as we show in table 7 Last but not least, I would like to write a resume about the costs of the hardware I have used for this research including the boards, wires, batteries, and my laptop. We will consider that a year has 220 work days a year, 8 hours a day, therefore cost per hour per device is

$$Device\_cost/lifespan * 220 * 8 \tag{3.1}$$

As we show in table 8 We estimate a lifespan of 4 years for the computer, and the boards can last about 10 or 15 years without maintenance. We consider that we charge the batteries one day every two days but we only use them in the data collection task.But to lifespan effects we calculate one cycle every two days. The result is 0.68 years. On the other hand the wires can last for 20 years in perfect conditions if they all well stored or even more.

3.2 Contingence Plan

As any other project in the IT environment, we must have a well defined contingence plan, with a calculation of the cost overruns due to obstacles or unexpected issues with the devoloping of the different parts of the work. Since we are dealing with technologies we haven't work before, the possibility of finding problems along the way to our goals is remarkable so we have made a forecast of an increasing of 15% over the total cost. To understand all this stuff a little bit better we have build up the table 9

ID	Task	Work Hours	Roles	Costs	Costs+SI
PM	Project Management	107h		4834€	6284.2€
PM.1	Scope of the project	15h	PM	390€	507€
PM.2	Time Planning	10h	PM	260€	338€
PM.3	Budget	4h	PM	104€	135.2€
PM.4	Sustainability Report	4h	PM	104€	135.2€
PM.5	Meetings	14h	PM,D,S,DVOPS,R,T	1666€	2165.8€
PM.6	Documentation	50h	PM,R	2050€	2665€
PM.7	Presentation	10h	PM	260€	338€
PW	Previous Work	30h		590€	767€
PW.1	State of the Art	10h	R	150€	195€
PW.2	Setting up of the work environment	20h	D	440€	572€
D	Development	340h		7480€	9724€
D.1	Development of the board	260h	D	5720€	7436€
D.1.1	Fix the protocol	80h	D	1760€	2288€
D.1.2	Connection to Wifi	80h	D	1760€	2288€
D.1.3	Transmit the logs	100h	D	2200€	2860€
D.2	Development of the server	80h	D	1760€	2288€
I	Integrating parts	80h		2240€	2912 €
I.1	Containerizing the server	40h	DVOPS	800€	1040€
I.2	Communication test	40h	DVOPS, T	1440€	1872€
DT	Data treatment	140h		3100€	4030€
DT.1	Data collection	20h	S,R	700€	910€
DT.2	Data preprocessing	60h	S	1200€	1560€
DT.3	Graphics	60h	S	1200€	1560€
	Total	697h		18244€	23718€

Table 6: SI stands for Social Insurance,Roles are Project Manager, Researcher, Developer,Statistic, Tester, DevOps.source: Own elaboration

Software	Cost per month	Months	Total Cost
Plattformio IDE	0€	5	0€
Overleaf	0€	5	0€
Gantter	5€	5	25€
Trello	0€	5	0€
Elastic Stack	0€	5	0€
Total	-	-	25€

Table 7: Software expenses,source:own elaboration

Hardware	Price	Units	Lifespan	Hours	Amortization
Laptop Acer Nitro 5	999€	1	4 years	697h	98,90€
Board TTGO T-Beam	37.95€	4	15 years	320h	1.84€
Wires	9.99€	4	20 years	300h	0.34€
Batteries	7.95€	4	0.68 years	20 h	0.53 €
Total	-	-	-	-	102€

Table 8: Costs from hardware resources,source: own elaboration

Type	Cost	Contingence
Place	1000€	150€
Software	25€	3.75€
Hardware	1223€	184€
Staff	23718€	3558€
Total	25965€	3895€

Table 9: Contingence table of 15% per type of cost,source: own elaboration

3.2.1 Unexpected issues

Finally we must have in mind that some problems may appear during the development of the project. The most likely things that can happen are troubles with the development of the boards trying to send data to the server, or maybe some kind of issues trying to get the boards connected to the wifi network . And these have a quite high probability to happen. In the following lines we will try to quantify how often this kind of issues may appear. At the end of the description in table 10 we have a list of every unexpected issue with its cost aside.

1. Increase of the boards' development time. Since these board work with an RTOS and they are absolutely unknown technology for me the chances of finding any kind of problem during the process of connection to the wifi, fixing the LoRaMesher or transmitting the data to the server are high, about a 20% of probability. The development time would increase 25 hours and probably that would mean an increase of the integration test time by 10 hours.

2. Increase of the server's development time .Though it will be developed in a technology I know a little, it may not work at glance. The chances of the server not working the way I planned are high, about a 25% and maybe it takes 20 extra hours of work plus 10 hours of integration testing.

3. Device failure. If the boards get wrecked I will need to get a new ones.

Unexpected issue	Cost	Risk	Total cost
Increase of board development time	710€	20%	142€
Increase of server development time	600€	25%	150€
Failure of laptop	1000€	10%	100€
Failure on board 1	37.95€	5%	1.89€
Failure on board 2	37.95€	5%	1.89€
Failure on board 3	37.95€	5%	1.89€
Failure on board 4	37.95€	5%	1.89€
Failure on libraries	820€	5%	41€
Total	3282€	-	441 €

Table 10: Costs overrun, source: own elaboration

Nevertheless, this would be quite strange.Its probability is around 5%. A little bit more probable is to have any trouble with my laptop,about 10% of chances,and that would be quite a big mess. Getting my laptop repaired could mean a whole week of delay on the tasks time planning. That's about 20 hours more at the task I would be developing.

4. Failure on the libraries. LoRaMesher is using a special radio libraries to transmit data packets, as well as to get connected to the wifi.Though this libraries have been tested and they are expected to work properly, there is always a small chance that things were not designed to work with your boards in the same way as they did the boards they did the tests with. It would take about 30 hours of extra work and 10 hours of integration testing. The risk of this to happen is around 5%

Finding Unexpected issues may cause a delay at the time of reaching our goals in this research, whether if it is a development time issue tha I would mean about a week of delay, or It's a failure on libraries or even a health issue. Some of these are easier to overcome, some of them are only a matter of time, some other may become a real problem and there will be no good enough alternative plans. In that case we could leave some of the sub goals aside or maybe change the point of view of the whole research .

3.2.2 Total costs

Once we have introduced all the costs of the project, in the table 11 we have written down the total cost of the project which is 30301€

Type	Cost
Place	1000€
Software	25€
Hardware	1223€
Staff	23718€
Contingence	3895€
Unexpected	441€
Total	30301€

Table 11: Total cost of the project,own elaboration

3.3 Control Management

Once we have defined the initial budget, we will define the control mechanism to avoid deviations as well as numerical indicators that help to the control of the whole thing. In the weekly meetings, every time a task will be finished, the budget will be updated with the real time of effective work comparing them with the estimated time.
In order to have the unexpected issues under control, when a task will be finished the additional expenses will be written down, and they'll be compared with the unexpecteds' forecast and the contingence plan. Following this methodology we will be able of detecting any deviation or doing a forecast about any task to need to be increased or reduced.
Now lets introduce the numerical indicators :

1. Deviation staff costs per task:
   $(estimated\_cost - realcost) * realworkhours$
2. Task execution deviation
   $(estimated\_work\_hours - real\_work\_hours) * realcosts$
3. Total Deviation on task execution
   $(estimated\_total\_cost - realtotal\_cost$
4. Total estimated costs of unforseen
   $unforseen\_estimated\_cost - unforseen\_real\_cost$
5. Total hours deviation
   $estimated\_work\_hours - real\_work\_hours$

3.4 Sustainability

In every project it's important to do a sustainability analysis having in mind the three dimensions: economic, social and environmental. In the coming lines the author of the thesis will do a self evaluation about the domain of the sustainability competence followed by an analysis of the three dimensions of the work based on a compendium of questions.

3.4.1 Self Evaluation

Along the Informatics degree we have been taugh to consider the environmental impact when you are about to start a project of any kind and what should we have in mind when we are talking about sustainability .In spite of being taught about sustainability It's at the point when you are about to start a thesis like this when you realize how essential this subject is.
From a very personal point of view, I have always considered the economical side of every enterprise I get into in order to ensure the viability since when I get enrolled into a project like this, we'll be looking for an economic improvement of an existent solution .
Nevertheless when we get to the point of analyzing the social point of view, we don't realize that much how essential this can be as well as the environmental side. Searching along the whole internet I have realized that the improvements in the IoT world must look after the environment and become also an improvement of the whole society as well.
For all these reasons I strongly think that having an usable and efficient communication protocol for the upcoming IoT world can allow us to make an easier to handle world which is one of the goals that technology was made for, improve the life of the people on earth with the less environmental damaged.

3.4.2 Economic dimension

Considering that knowing the performance of this new protocol could bring us a lot of benefits in the field of IoT.
Since until now there has been only solutions using single channel transceivers with an iterative algorithm using different spread factors but none of this was thought for multi hop algorithms, I think that a thesis to study the real possibilities of a new algorithm for less than 35000€ is a very fair cost. And IoT and pervasive computing is one of the most pushing technologies in the present, so any improvement in the field will have lots of applications on the industry, the medicine or the transport.

3.4.3 Environmental dimension

The elaboration of the boards to deploy this protocol have an environmental impact, just like any other electronic device just like the laptop I'll be using to develop the rest of the parts of the project. Hence we can not avoid using them since we are in the Informatics degree and there will be a hardware to work with despite all the environmental impact that it implies

On the other hand the uses of this boards may help to track helds of animals in danger of extinction, or maybe sea mammals among many other uses caring for our environment wild life which is something really priceless.
All the meetings will be held online, but I could do them in person due that my home is 5 minuts by walking to the upc and for me It wouldn't mean further problem. To carry on the whole thesis, the environmental impact will be as minimun as possible to this kind of enterprise.

3.4.4 Social dimension

Having an usable implementation of this new communication protocol and knowing how good it works, may become in something really useful in our society. The multiple uses of the IoT in our daily tasks in the industry world is unending.

The thing that catched my eye the most of this project was the data treatment,considering that statistics was a field which I always wanted to know deeper, and that It was about a protocol whose idea was from someone of the UPC I have a colleague relationship with since 2007, which for me was kind of significant meeting us again in this circumstances.

The project was born from the needs of quantifying the well working of this LoRaMesher multi hop new made up protocol, bringing it from the theoretical idea to the final implementation.

3.5 Legal considerations

3.5.1 LoRa

We have to be careful at the time we will be gathering data from the LoRa Mesh network status . And it's not about how many post request we are sending towards the server since the spectrum for the wifi we will be using is free to use for private goals, but the LoRa data that the boards we will be sending among the different nodes has to follow certain rules in order to follow the current spanish legislation

Region	Frequency(MHz)
Asia	433
Europe, Rusia, India, Africa	863-870
US	902-928
Australia	915-928
Canada	779-787
China	779-787,470-510

Table 12: LoRa Frequencies by Country. Source: "Frequency Plans by Country", 2021[11]

about usage of the radioelectric spectrum . Lets take a quick look at legislation set up by the European Union in the case of Spain as we show in table 12 and in other regions of the world .
Along all the experiments that we performed testing the monitoring system that we finally built, not only we had to consider the Industrial Scientific and Medical band that our LoRa modules would be working in, but the percentatge of the time that we will be occupying during our long-term transmissions on a given channel in order to meet the local laws requirements in order to not break the law referring to spectrum usage for these kinds of scientific goals. In this particular case, in Spain we are only allowed to transmit a 1% of the duty cycle although we could avoid this using the Channel Activity Detection(CAD)[26]. The frequence of the LoRa boards is given by the implementation of the LoRaMesher library, but the usage of the duty cycle depends on the configuration which we flashed every board to execute the experiments and gather the data. Maybe for small experiments we can be more flexible with the legal considerations, but for long-term monitoring experiments we must be careful with these kind of things and be fair with the spectrum usage.

3.5.2 General Data Protection Regulation

We will be doing experiments in a controlled environment all the time and we are not sending personal sensible information and our post requests towards the node server are being sent in plain text since we are only doing experiments with a raw implementation of the protocol and the boards are not binded to any person or entity. If wanted to monitor in real time the behaviour of the protocol in a real usage case over the field we would have to be very careful whether we will be gathering data from a board which only belongs to someone or some kind of institution . For that task we have already done some researchs on the Elastic Stack in order to have a TLS security layer in order to avoid third people to read

the content of the monitoring packets in order to meet the European directive "General Data Protection Regulation" “Official Legal Text”[21], but by the time this Bachelor Thesis was made, a full production version was not acomplished yet but nearly done.

3.5.3 Intelectual Property

In the very beginning of the project, the initial idea was to make everything open source, since there is not much code whithin.It has been more a matter of joining different software and hardware components together and make them work. However at the end of the thesis I was hired as research support tecnician to go on with this work, and I have some clauses that obly me to not share any of this material to third people without the permission of the UPC. So any license that this code may have must be first agreed with the UPC.

Chapter 4

Analysis of the LoRaMesher library

Lets begin with the inner structure of the LoRaMesher library for the process and packets management. Joan Miquel build a clear structure of priority queues to handle all the packets travelling among the TTGO-Tbeam boards. For that reason he wrote down a template with the following structure in list 4.1 .

```
1 class PacketQueue {
2 packetQueue < uint32_t >* first = nullptr ;
3 public :
4 void Add( packetQueue < uint32_t >* pq) ;
5 template <typename T>
6 packetQueue <T >* Pop () ;
7 size_t Size () ;
8 void Clear () ;
9 };
```

Listing 4.1: LoRaMesher Priority Queue template source : Joan Miquel Solé's Bachelor Thesis

Made out of this template, he built up three different kinds of queues for three different objectives :

1. Received queue : this queue is for the received packets in order to not miss any received packet they are put in this queue.
2. To send queue: this data structure holds all the packets that are meant to be sent.
3. Received user packets: this queue is the place where we place all the packet for the user that we receive from the process packet routine,this adds it to its priority queue.

Then we have a number of routines that run among all the cores with specific goals ech one. These are the following

- On receive. The highest priority process. It's supposed to be very light in order no not lose any packet, it takes the packet and encapsulates it in a packet queue and puts it into the queue of received packets.
- Send Hello Packet. This routine activates every 30 seconds creating a packet and encapsulating in a paquet queue and puts it in the to send queue.
- Process packet. With this routine packets are processed, and it gets activated every time a notification from the On receive routine arrives. Process the first packet to arrive and it's in charge of the packet forwarding and of the routing protocol putting all the packets in the to send queue. Also handles other protocols. This routine might be very useful for our goals.
- Send Packets. This routine sends packet every x time. It's crucial for the duty cycle and to acomplish the requirements of the wavelength that it's supposed to have to follow the laws of the legislation of the European standards.
- Receive users data. This routine receives a notification every time a node receives a packet for the user of that node, basically it is a loop waiting for notifications. It 's implemented by the users

When a signal is created, the function that receives it gets maximum priority, the packet is created and gets enqueued on Received queue that sends a signal to be processed. This concrete process can be paused at any moment for the packets to be keeped on arriving.

4.1 Types of packets

In the same way we must consider the three kinds of packets that LoRaMesher has . On one side we have the generic packet showed in list 4.2

```
1  # pragma pack (push , 1)
2  template <typename T>
3  struct packet {
4  uint16_t dst ;
5  uint16_t src ;
6  uint8_t type ;
7  uint8_t payloadSize = 0;
8  T payload [];
9  };
10 # pragma pack (pop)
```

Listing 4.2: Generic Packet source Joan Miquel Solé's Bachelor Thesis

One of the smartest points of this declaration are the pragma instructions that will align the size of the data structure to the architecture of the memory of the board to fill the gaps. In this case is everything aligned to one byte size. At the end of the declaration the alignment returns at its original size.
Coming up in the list 4.3 we have the second type of packet. Lets have a look at it.

```
1 struct networkNode {
2 uint16_t address = 0;
3 uint8_t metric = 0;
4 };
```

Listing 4.3: Routing packet,source Joan Miquel Solé's Bachelor Thesis

The total size of this packet is 3 bytes, it contains an address and a metric. And finally we will describe the data packet which has the following structure in list 4.4 :

```
1 # pragma pack (push , 1)
2 template <typename T>
3 struct dataPacket {
4 uint16_t via ;
5 T payload [];
6 };
7 # pragma pack ( pop )
```

Listing 4.4: Data Packet, Joan Miquel Solé's Bachelor Thesis

It's also aligned to 1 byte and follows the same pragma structure as the generic packet. It contains a via field, and payload that can be as big as 248 bytes, since He is using two bytes for the via attribute.
The last thing we must consider to start working on it will be the structure of the packet queue showed in list 4.5 where the packets will be encapsulated to be stored, sent or put in any of the queues.

```
1 # pragma pack (push , 1)
2 template <typename T>
3 struct packetQueue {
4 uint32_t timeout ;
5 uint8_t priority = DEFAULT_PRIORITY ;
6 LoraMesher :: packet <T >* packet ;
7 packetQueue <T >* next ;
8 };
9 # pragma pack ( pop )
```

Listing 4.5: packetQueue structure Joan Miquel Solé's Bachelor Thesis

The remarkable fields are the timeout that will be used to discard the packet when finishes, the priority that tell us whick packet will be send, processed or stored first, remembering that the higher value, higher the priority. Following the above

described logic we have build the design of what will be the monitoring system, trying no not interfere the correct working of the LoRaMesher.

4.2 FreeRTOS

All the code was scheduled by the FreeRTOS, for this reason Joan Miquel created an individual task for every routine, as an example of it I will write down in list 4.6 the implementation of the highest priority task in my comrade code:

```
1 int res = xTaskCreate(
2     [](void* o) { static_cast<LoraMesher*>(o)->receivingRoutine();
      },
3     "Receiving routine",
4     4096,
5     this,
6     4,
7     &ReceivePacket_TaskHandle);
```

Listing 4.6: Receiving routine task implementation, source: Joan Miquel Solé's Bachelor Thesis

In this 4.6 task we can see several parameters. The first one is the routine that will be executed within the task, Receiving routine is just a name for the task, 4096 is the number of words (not bytes!) of the given task, this is the pointer to the parameters passed into the task, 4 is the priority of the task, in this case the task designed for receiving packets is the highest priority task. The last parameter is a pointer to the taskhandle, this will be very useful in the destructor of the class where we take the tasks out of the FreeRTOS scheduler. Besides this task Joan Miquel build 4 more tasks, each task with a lower priority from 3 to 0 in the following order : Sending routine, Hello Routine ,Process routine and Receive user routine. . By default, FreeRTOS uses a fixed-priority preemptive scheduling policy, with round-robin time-slicing of equal priority tasks:

- "Fixed priority" means the scheduler will not permanently change the priority of a task, although it may temporarily boost the priority of a task due to priority inheritance.
- "Preemptive" means the scheduler always runs the highest priority RTOS task that is able to run, regardless of when a task becomes able to run. For example, if an interrupt service routine (ISR) changes the highest priority task that is able to run, the scheduler will stop the currently running lower priority task and start the higher priority task - even if that occurs within a time slice. In this case, the lower priority task is said to have been "preempted" by the higher priority task.

- "Round-robin" means tasks that share a priority take turns entering the Running state.
- "Time sliced" means the scheduler will switch between tasks of equal priority on each tick interrupt the time between tick interrupts being one time slice. (The tick interrupt is the periodic interrupt used by the RTOS to measure time.)

Joan Miquel avoided starvation of the lowest priority tasks because the highest priority tasks is waiting for a notification with the following 4.7 instruction and gets blocked:

```
1 TWres = xTaskNotifyWait(
2       pdFALSE,
3       ULONG_MAX,
4       NULL,
5       portMAX_DELAY // The most amount of time possible
6     );
```

Listing 4.7: Receiving Routine waiting implementation,source: Joan Miquel Solé's Bachelor Thesis

xTaskNotifyWait() waits, with an optional timeout, for the calling task to receive a notification. If the receiving RTOS task was already Blocked waiting for a notification when the notification it is waiting for arrives, the receiving RTOS task will be removed from the Blocked state and the notification cleared. While this routine is blocked, gives space to lower priority routines to reach the processor.
The second place in terms of priority is the sending routine, but this routine is sleeping for 10 seconds so it will not be ocupying the processor all the time. The following routine in priority terms is the process routine. This routine is also waiting for a notification, but uses the instruction we show in list 4.8 which differs from the previous one.

```
1   ulTaskNotifyTake(pdPASS, portMAX_DELAY);
2
```

Listing 4.8: process routine mutex implementation,source:Joan Miquel Solé's Bachelor Thesis

This is the standard form that FreeRTOS has of implementing a mutex. When entering at the routine it takes the mutex until a notification is arrived.
And then the last routine in the priority scale that will have to wait to everyone to be blocked or waiting will be the receive users routine.

Chapter 5

Design of the monitoring system

For the monitoring system we have considered using some extra data structures and a couple of routines that will enqueue the packets to be send towards the server and will unwrap the packets and get them ready to be sent through a post request into different mongo models in our containerized server. But afterwards we changed our mind and decided to do it as simple as possible, at least for this first fully functional version. We will leave it as a simple task with a very simple routine that will grab the processor every 30 seconds and will perform a post request. Losing one of this messeges is not critical since all the info contained in one message will be as well in the next one, so having a queue to send every message towards the server would be counterproductive because it will only make having older info in the server in case of packet loss and we want to have the newest one. So if a post request for any reason doesn't reach its destiny, don't panic, wait for the next one, the info will take a little bit longer to get updated but it will be finally fine.

5.1 Keeping up FreeRTOS stability

All the inherited code from Joan Miquel thesis is built over a built-in FreeRTOS system as we told before.Considering this we have only one more routine to add in the scheduler and this is a clear advantatge comparing to other possible solutions since we want to monitor the node traffic with the minimum overhead in the scheduler. We have an aproximate idea of which priority we must assign to our new routine. The added routine must get blocked or be waiting to leave processor space to lower priority routines to keep on executing, as well as the routines with the same priority reach the processor. Since the receiving routine is the most prioritary we will leave it as it is because losing packets is critical for the good working of the protocol. We dismiss as well the possibility of sharing priority with

sendPackets routine because we think that this routine must not be interrupted if possible. We have four main options left:

1. Leave the added routine at the same priority as send Hello Packet routine, the priority number 2 and leave the scheduler to do its job between the two tasks aiming for their piece of processor time. Send Hello packet routine is not such a time dependent routine and its working it's not that critical for the whole protocol at the proper time.

2. Leave the added routine one step below in the priority scale, the number 1, shared with the process packet routine. We could have a try setting the new routine to this priority and see whether we see any improvement.

3. Set the priority to 0 just like the receive user routine. Lower priority tasks may be in danger of inanition.

4. The last chance would be shifting all the priorities one position, and place our new routine somewhere below the second highest priority.

To find the best priority fit we will have to do some experiments and observe if there's any packet losing event or any task that doesn't reach the processor properly. But due to the lack of time to do further researchs, we will start setting the priority for the new task at two, observe the behaviour and if there's some extra time do some experiments considering other priorities.

5.2 Options to build the data flow to the server, client side

To send the data from the boards towards the server we considered a couple of libraries with these fucntionalities. Since we are working with FreeRTOS we heard about the built it library of RTOS designed to establish an http connections towards a server. This library is called corehttp.

5.2.1 coreHTTP

This is supposed to be the built-in library of FreeRTOS to build requests from an IoT devices running this RTOS . It provides a subset of the HTTP 1.1 standard.The HTTP standard provides a stateless protocol that runs on top of TCP/IP and is often used in distributed, collaborative, hypertext information systems. This library has been optimized for a low memory footprint. The library provides a fully synchronous API to allow applications to completely manage their concurrency.

coreHTTP also operates only on fixed buffers, so that applications have complete control of their memory allocation strategy providing a high-level simple API to serialize request headers, send the request, and receive the response. It is decoupled from the underlying network drivers through a two-function send and receive transport interface[4]. The application writer can select an existing transport interface or implement their own, as appropriate for their application.The library has proofs showing safe memory use and no heap allocation, making it suitable for IoT microcontrollers, but also fully portable to other platforms.This library can be freely used and is distributed under the MIT open source license.
Nevertheless, despite all these functionalities and optimizations, we found some problems when we tried to use it. We will explain briefly what was all about when using this library.

- The library has a very complex sintax. It's meant to be used at a very high explotaition of resources from the boards and to have a very good performance in production for professional applications but this adds a thick layer of using complexity .
- The using examples are quite scarce, and they were all designed to work with aws, another technology wich I don't know. It's possible to use a server of your own but the documentation I found was really unclear for the few data I wanted to transmit and It was too difficult to adapt.
- All the documentation was specially focused on a windows environment and I do work with a linux environment. Changing of environment or trying to find the right info for Linux and PlatformIO was absolutely out of reach.
- In general terms it was a library full of features but too difficult to read, understand and use for the home data transfer system that we wanted to build and I had already a lot of job done from the server side and it was too difficult to extract only the most interesting functionalities, and leave aside all the rest.

For all these reasons we had to discard this data transfer library and look for better and more simple options to take a few data out of the TTGO-TBeam boards which is what we want to do for this thesis.

5.2.2 Arduino's HTTPClient

HTTPClient is an arduino module specially thought to execute simple http requests. Comparing to the previous library, this one has a lot of advantages.

- Very easy reading and using library

- Very well documented and plenty of examples with a huge community
- Very flexible and easy to adapt to previous code

Maybe this library is not so well optimized but we will do our best to enhance his performance working together with the LoRaMesher, and taking advantage of its simplicity, we had a data flow between our node server and the boards in a very little time.

5.2.2.1 Using the HTTPClient

To perform an HTTP POST request we needed very few steps.

1. We take the address of the endpoint to open the socket towards the server
2. We perform the socket connection with the code on 5.1
3. We add some headers to we add some headers to the POST request to determine whether are we going to send, a JSON or urlencoded data
4. We prepare the buffer with the content of the request with the given values with an sprintf
5. We execute the post request.
6. We close connection

```
1 micliente3.begin(host3.c_str());
```

Listing 5.1: Socket connection from Network.cpp,source own creation

Post request data flow format Just like any other web application, in this case we had two ways of sending data towards the node server. We could have sent a JSON instead of sending a URL enconded request but we decided by this last way for simplicity. JSON was also a choice to consider, but after a few tries we saw that was more prone to show off panic core errors, which means that there was a buffer overflow or a segmentation fault somewhere in the memory so we finally decided that we will do it the simplest way for this thesis. We had to be careful also doing it like this because there's also a buffer limit to perform the request and we had a lot of fields to send. We had to shorten as much as possible the length of the request and the name of the fields, otherwise the boards started to reboot spontaneously and that was an unwanted behaviour. Sending JSON had also had another problem because there are two ways of sending JSON, with a fixed size buffer, or dinamic size buffer. Everything put together just added complexity to the working of the whole system so by the moment we decided to do it the other way and discard sending JSON .

5.3 Wifi connection

To transfer the data between the LoRa boards and our server side we needed to have an stable Wifi connection. We didn't have many choices to do that with an ESP32 environment so we choosed the simplest one, the one coming with the arduino framework called WifiClient that executes a simple wifi connection with a ssid and a password as only parameters. The setup with the platformio might become a little bit tricky when you are not used to. But the code and the theory of the connection has no further complications as we see from the code in list 5.2

```
1  Serial.print("Connecting to ");
2      Serial.println(ssid);
3      WiFi.disconnect();
4      WiFi.begin(ssid, password);
5
6      while (WiFi.status() != WL_CONNECTED) {
7          delay(500);
8          Serial.print(".");
9      }
10
11     Serial.println("");
12     Serial.println("WiFi connected");
```

Listing 5.2: wifi connection with WifiClient,source:own creation

5.3.1 Our Network LoRa Module

Using the previous libraries we have built up a Network module in charge of doing the Wifi connections from every board to the internet, and sending the post request every 30 seconds with the updates of the state of the incoming and outcoming packets of each kind. It is in charge of updating all the packet counters with public methods that will be called from the LoRaMesher code every time a packet is received or put in the queue to be send.So the Network module is minimally invasive inside the LoRaMesher code,only a few calls to these public methods 5.3 from the Network module in the proper code line numbers. Lets make clear with a brief description of every counter update method.

- Send Packets are the total number of packets that are being sent including data and routing packets
- Rechellopackets are total of the received hello packets in the system
- Broadcast are the received packets which destination is the broadcast address. Remember that data packets are sent via broadcast and only the node shall keep the ones that has its destination the node itself.

- Forwarded packets are the received packets that will be forwarded to other node, that means that I have the destiny inside my routing table and the packet will take its way .
- data packet for me are the number of received data packets which destiny is this node and I should keep.
- I am via are the counter that keeps the record of the received packets that are data packets and the via is this node.
- Destiny unreach are the packets that I will delete because the destiny is unreachable.
- Sent hello packets, as its name says are the routing packets that are sent by this node.
- The last counter are the data packets that I receive but they are not for me so I shall delete them. This means that these packets are not any of the previous kinds of packets.

```
1         void incReceivedPackets();
2         void incSendPackets();
3         void incRecHelloPackets();
4         void incBroadcast();
5         void incForwardedPackets();
6         void incdataPacketforMe();
7         void incIamVia();
8         void incDestinyUnreach();
9         void incSentHellopackets();
10        void incNotForMe();
```

Listing 5.3: Details of network.h,source:own creation

The next important thing is encapsulate this routine in a task to be inserted in the scheduler with the code in 5.4

```
1 res = xTaskCreate(
2     [](void* o) { static_cast<Network*>(o)->sendData(); },
3     "Sending data Routine",
4     4096,
5     this,
6     2,
7     &SendingData_TaskHandle);
```

Listing 5.4: Network module main task,source: own creation

The creation of the task had no further problems since I only had to follow the examples on Joan Miquel code of task creating process. It has only a pointer

to a data sending routine that sends info every 30 seconds, a size in words, and a priority that we set to 2. We thought that 2 would be a good priority for the beginning, of course this is not an absolute value, but we thought that this priority couldn't be much higher because the higher priority tasks are in charge of receiving packets and send packets, so we thought that this routine could share priority with the hello packet sender routine because it's not a time critical routine because interfere with higher priority routines may give as a result packet loss issues which is not desirable. Nevertheless we will make some experiments to ensure the correct working of the whole system.

5.4 Server side

First of all we needed to develop a server. We decided to use Node for no particular reasons. One of them was my previous experience in the field, but we could have used another language or framework, just like Python with Django, Java, php or whatever. But in this case we will do it with node.

5.4.1 Node.js

We developed a node server taking a project from the AC department as an example. The project was called Cow Localizer, a work developed over boards TTGO-Tbeam to gather data about cows' coordenades to a node server. We added some more models, some more controllers,and some more routes to the server to adapt them to the data we wanted to collect. In addition we added some exception handling events for a better testing .

5.4.2 Mongo

Since the project I took as an example was already implemented using mongo as a database, the straight way was keeping the same logic to add my models to the server. It's a very well known NO-SQL database and works very well along with node thanks to mongoose, the library that connects the node environment and the mongo database. It's very flexible and scalable, widely used for plenty of goals worldwide, and also will suit for this thesis. We could have used a file as a persistence, but having almost all the server written down, the decision was already taken.

5.4.2.1 Dockerizing the server

To enhance the portability and usability of the server we invested a little time dockerizing the server and the database wich is something that the team of the

CowLocalizer project didn't do. The main issue doing this was that we needed to dockerize the node server and the database separately. The dockerfile of the node server was straight away as we show in list 5.5

```
FROM node:carbon as builder
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["npm","run","dev"]
```

Listing 5.5: node Dockerfile,source: own creation

It is based on a prebuild image of node took from the web page dockerhub using a very light version of linux called carbon with the minimum features to run node. To run both containers and establish a communication channel between them I had to use docker-compose. The process of developing a docker-compose file, with persistence layer, was a little bit tricky but I finally did it the right way, the file goes like this docker-compose we show in list 5.6

```
version: "3"
  #docker run -p 8080:8080 cowserver:1.0

services:
  web:
    container_name: myapp
    restart: always
    environment:
      LORA_MONGODB_HOST : "mongo"
      LORA_MONGODB_DATABASE : "lora-app"
    ports:
      - "8080:8080"
    build: .

    links:
      - mongo
  mongo:
    container_name: mongo
    image: mongo
    volumes:
      - ./todo-mongodb-data:/var/lib/mongodb
    ports:
      - "27017:27017"
```

Listing 5.6: node server docker-compose.yml,source:own creation

We see both containers, the web server, and the prebuild mongo container . In volumes we see the path inside the container were the data will be stored and the communication channel between them.

5.4.3 Monitoring tools

Since we are aiming to extract some graphics out of the metrics within the LoRa boards we started thinking about what would it be the best way of getting the data out of the board and plot the graphics. A very simple way would be doing a Mongo export to a csv file and then using a high level language such as R o python to visualize the graphics.This was the initial thing we planned. But doing so we would have to do all the process manually . So we thought about a way of doing that without any human action between which also was scalable and easily customizable to new kinds of data and metrics, and able to monitor a network status in real time . That's why we thought that wouldn't it be great if we had a data server that allowed us to visualize all the data among the nodes using the node server as a middleware, where the data flowed automatically towards the data server in order to plot any kind of graphics in a very usable and simple way, being able to gather big amounts of data in long experiments without not even worry about extracting the raw data from the node server to the data center ? That's when we heard about several tools that could help us to do this. One was the well known tool called Grafana and the other one is called ElasticSearch. We will make a brief description of both of them and finally we will explain why we have chosen Elastic Search for this concrete task.

5.4.3.1 Monitoring mesh networks:state-of-art

This won't be the first time that a mesh network is being monitored.
In the last decade, mesh networks have been drawing considerable attention from operators and service providers due to their potential for extending the coverage of public hot-spots, corporate buildings or large-scale urban areas; enabling savings on cabling, deployment and maintenance costs. In this case, mesh networks represent a continuation of the fixed/wireless networking infrastructure (Core and Radio Access Network), with users being expected to demand similar services, e.g., browsing, email, multimedia computing, collaborative networking applications. From a network management perspective, the challenge related to the deployment of mesh networks lies in providing effective service provisioning despite the unreliability caused by unpredictable addition, failure or removal of Network Elements (NEs), and also network merging or partitioning. This requires provision of adequate support for topology monitoring so as to enable applications, and/or network administrators, to react in a timely way to any topology change. Note that network monitoring is not restricted to this usage. On the contrary, network monitoring may also be used for (i) providing statistics to pinpoint the sources of network failure, (ii) verifying if the strategies adopted by (routing) protocols, applications or middleware perform well, and (iii) locating potential bottlenecks

so as to redimension the network. Traditional solutions for monitoring wire-line networks provide poor performance in mesh networks due to several reasons. First, unlike wired network, wireless mesh networks are characterised by the absence of underlying infrastructure; the network being maintained by the combined effort of the constituent hosts. Second, these hosts often operate under severe constraints such as limited bandwidth for example. Third, mesh networks experience significant signal quality fluctuation caused by unreliable physical medium, obstacles, interferences, hidden hosts and some varying conditions in the environment [24] Previously researchers proposed a monitoring architecture based on a Topology Controller (TC). A TC corresponds to a cluster head which is responsible for building and maintaining a local view of its cluster(s) and the logical connections between itself and the neighbouring cluster heads. A TC further acts as a mediation point and may also aggregate and correlate data for the cluster. TCs co-operate with each other to build and maintain the network topology (i.e., the entire network, or a particular level or a sub-tree of the topology). A full topology consists of an aggregation of all local topology information for the TCs in that topology. The monitoring service gathers measurement information from all, individual, or even components of NEs. When the service on a NE receives a request for information, it passes the request to all its subordinate nodes in parallel. Each subordinate node, in turn, passes the request down the tree until the bottom of the tree is reached. The data is then read by each node and passed up to the superior nodes where it is aggregated and again passed upwards until the top of the tree is reached.
Nevertheless, concerning to LoRa mesh networks there's not much work about monitoring systems gathering info using software tools, extracting data automatically directly from the LoRa devices via http which is what we are going to try in this thesis .

5.4.3.2 Grafana

Grafana open source software allows you to query, visualize, alert on, and understand your data no matter where it's stored. With Grafana you can create, explore and share all of your data through elegant, flexible dashboards [16] This solution has several interesting features but Its purpose is slightly different than the ElasticSearch one.Among them we have the following[5]:

- Unify your data, not your database
  Grafana doesn't require you to ingest data to a backend store or vendor database. Instead, Grafana takes a unique approach to providing a "single-pane-of-glass" by unifying your existing data, wherever it lives.

  With Grafana, you can take any of your existing data- be it from your

Kubernetes cluster, raspberry pi, different cloud services, or even Google Sheets- and visualize it however you want, all from a single dashboard.

- Dashboards that anyone can use
  Not only do Grafana dashboards give insightful meaning to data collected from numerous sources, but you can also share the dashboards you create with other team members, allowing you to explore the data together.

  With Grafana, anyone can create and share dynamic dashboards to foster collaboration and transparency.

- Data everyone can see
  Grafana was built on the principle that data should be accessible to everyone in your organization, not just the single Ops person.

  By democratizing data, Grafana helps to facilitate a culture where data can easily be used and accessed by the people that need it, helping to break down data silos and empower teams.

- Flexibility and versatility
  Translate and transform any of your data into flexible and versatile dashboards. Unlike other tools, Grafana allows you to build dashboards specifically for you and your team.

  With advanced querying and transformation capabilities, you can customize your panels to create visualizations that are actually helpful for you.

Besides all these features it has integration with several databases, including ElasticSearch. Being this last tool a more powerful and complete solution we finally decided that we would set up an elastic environment to gather all the data in the same place, and we will keep the mongo initial database as a backup of all the data. Doing so we will have not only the plots and the dashboards centralized, but the data itself in a distributed search engine. To sum up it allows a future integration with grafana if needed to.

5.4.3.3 Our proposal : ElasticSearch

Elasticsearch is a distributed, free and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured and non structured . ElasticSearch has been developed out of Apache Lucene and was introduced by first time in 2010 by ElasticSearch N.V. (now known as Elastic). It has a simple API REST, it's distributed, it's fast and scalable and It's the main component of the Elastic Stack, a set of tools designed for the ingest,the enrichment, the storage, the analysis, the visualization of big amount of data[3] . Commonly known as ELK

Stack (standing for ElasticSearch, LogStash, Kibana), includes a great collection of light agents known as Beats to send the data to ElasticSearch. In order to not discard the already developed node server to gather the data, we will take advantage from it and we will try to establish a data flow between the node server and the ElasticSearch engine. For that task we will be using the Javascript client for ElasticSearch

5.4.3.4 Kibana

Kibana is a free and open front end application that sits on the top of the Elastic Stack, providing search and indexing capabilities for data indexed in ElasticSearch [10] . We will be using Kibana as the user interface to monitor and manage and ensure our cluster of the Elastic Stack . It was developed in 2013 by the community of the Elastic Stack . It will allow us to plot any graphic we need through dashboards. Or even would help us loading big amount of data through LogStash if we weren't unable to establish a data flow between the node server and our search engine.

5.4.4 Dockerizing our Elastic Stack

When we decided to start using Elastic Search, we found ourselves in a dilemma. Should we make a raw installation over our system or shouldn't we find a quicker and more simple solution ? Installing new Software in our Linux is always risky, it is our main work tool and we need it to be stable as long as possible or we could get in troubles . So we decided to download a dockerized version of the Elastic Stack, and It was quite easy to find. After some searchs, we found a docker-compose file that could be useful for our goals , we show it in the list 5.7 .Since we have begun to know about how docker worked, adapting it to our own purposes would be now something affordable.

```
1  version: '2.2'
2  services:
3    es01:
4      image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
5      container_name: es01
6      environment:
7        - node.name=es01
8        - cluster.name=es-docker-cluster
9        - discovery.seed_hosts=es02,es03
10       - cluster.initial_master_nodes=es01,es02,es03
11       - bootstrap.memory_lock=true
12       - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
13     ulimits:
14       memlock:
```

```
15          soft: -1
16          hard: -1
17      volumes:
18        - data01:/usr/share/elasticsearch/data
19      ports:
20        - 9200:9200
21      networks:
22        - elastic
23
24    es02:
25      image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
26      container_name: es02
27      environment:
28        - node.name=es02
29        - cluster.name=es-docker-cluster
30        - discovery.seed_hosts=es01,es03
31        - cluster.initial_master_nodes=es01,es02,es03
32        - bootstrap.memory_lock=true
33        - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
34      ulimits:
35        memlock:
36          soft: -1
37          hard: -1
38      volumes:
39        - data02:/usr/share/elasticsearch/data
40      networks:
41        - elastic
42  es03:
43      image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
44      container_name: es03
45      environment:
46        - node.name=es03
47        - cluster.name=es-docker-cluster
48        - discovery.seed_hosts=es01,es02
49        - cluster.initial_master_nodes=es01,es02,es03
50        - bootstrap.memory_lock=true
51        - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
52      ulimits:
53        memlock:
54          soft: -1
55          hard: -1
56      volumes:
57        - data03:/usr/share/elasticsearch/data
58      networks:
59        - elastic
60
61    kib01:
62      image: docker.elastic.co/kibana/kibana:7.17.1
63      container_name: kib01
```

```
64      ports:
65        - 5601:5601
66      environment:
67        ELASTICSEARCH_URL: http://es01:9200
68        ELASTICSEARCH_HOSTS: '["http://es01:9200","http://es02
      :9200","http://es03:9200"]'
69      networks:
70        - elastic
71
72  volumes:
73    data01:
74      driver: local
75    data02:
76      driver: local
77    data03:
78      driver: local
79
80  networks:
81    elastic:
82      driver: bridge
83      external: true
84      name: custom_network
85
```

```
      84,4          Final
```

Listing 5.7: Elastic Stack docker-compose,first version,source:own creation

Notice that we have a little cluster of ElasticSearch made out of 3 elastic nodes, es01,es02,es03, all exposed at port 9200, all with its layer of persistence. And finally we have a service where Kibana works exposed at port 5601. All the services are sharing the same network called elastic that will be using the bridge driver. One more important thing. Since we have another two containerized services, the node server and the mongo database, we decided to make the network external with the name of custom_network in order to establish a communication channel through this network. Therefore we will have to modify the previous docker-compose file removing the link directive and replacing it by the networks directive and we will be adding these last lines in list 5.8 to the docker-compose file:

```
1  networks:
2    myapp:
3      driver: bridge
4      external : true
5      name: custom_network
```

Listing 5.8: network docker-compose parameters

Notice that our new network in this file will be called myapp. If there is any connectivity problem possibly because the external network needs to be created by running the command in list 5.4.4

```
1 docker network create -d bridge custom_network
```

The rest of the parameters are defining which are the master nodes, which are the hosts, the name of the cluster among others.

5.4.5 The Javascript Elastic client

In order to make the connection between our node server and our search engine, we could do two things. The first would be sending an average POST request towards our ElasticSearch engine with the data in JSON format. Since ElasticSearch indexes the data in JSON format that would be absolutely possible and there would be no need of using further techniques . Nevertheless there is available a special ElasticSearch client specially developed for using in JavaScript. We will make a brief description of its inner working to understand how it manages connections and asyncronous callings to reach scalability and enhance performance. This client has the following features

- One-to-one mapping with REST API.
- Generalized, pluggable architecture
- Configurable, automatic discovery of cluster nodes.
- Persistent, Keep-Alive connections.
- Child client support.
- TypeScript support out of the box.

Besides this, we will have a look to the inner schema of the client in the fig 4
There's an API available tha communicates directly with the transport layer where the serialization and the parsing of the data takes place, and finally we have a connections pool, one connection for every elastic node that we have available, where the selector would choose a connection from the connection pool when it gets the signal from the transport layer, it gives the connection to the transport layer, transport runs the connection in order to send the request, and moves it to the alive connections pool if it's successful otherwise it will be moved to the dead pool in order to retry the connection over time just like we see in the fig 5
Thanks to the API there's only one more step to get our data indexed into our elastic search engine, and this can be easily done from our node server with this

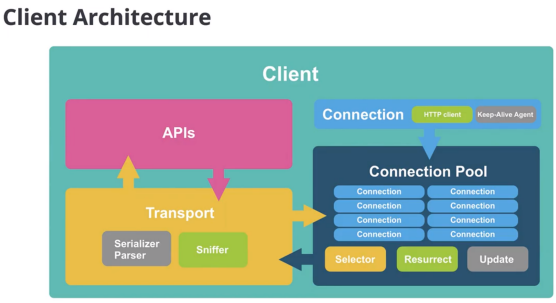


Figure 4: Javascript ElasticSearch client working schema, source:Elastic site

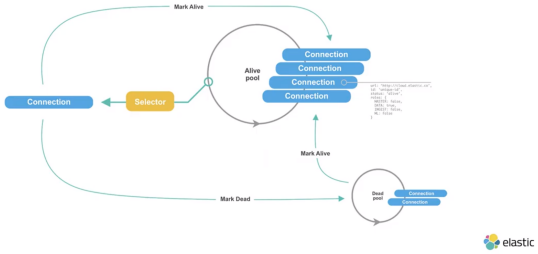


Figure 5: Connections pool working schema,source:Elastic site

5.9 piece of code. This has several advantatges because with this middleware we can manipulate data before ingest it into the engine, create new variables or do whatever we need, any kind of calculus and leave to the minimum the overhead between the boards and our node server. Of course there are other ways of doing this, like a tool called pipelines available in the elastic environment that would be like a kind of data pre processing, but this prevents us from learning further features from the elastic engine if we are more used to node programming and going straight forward.The most tricky issue of this call to the API was the local address field because we wanted to keep it in hex format and I had no I idea of what I needed to do that but I finally found the right way to do it as wee see at line 15 in list 5.9

```
1 await Client.index({
2     index: 'monitorization3',
3
4     body: {
5       recpackets: rp,
```

```
6         sendpackets: sp,
7         rechellopackets:rhp,
8         sendhellopackets:shp,
9         datapackme:dpm,
10        broadcast:brd,
11        fwdpackets:fwd,
12        dstinyunreach:dst,
13        notforme:nfm,
14        iamvia: ivi,
15        localaddress:ladd.toString('utf-8'),
16        totalreceived:totalreceived,
17        senddatapackets:senddatapackets,
18        timestamp:today
19
20      }
21    })
```

Listing 5.9: PacketTraffic.controlers.js details, source :own creation

5.4.6 Integration of the different parts of the server

First of all lets place all the different docker-compose files in the same file to set a quick and easy to use environment . This is not needed but we will do it like this because otherwise we have to keep on using the external network parameter. The bad side of doing this is the external network needs to be created first to have the whole system running otherwise there will be no communication between the containers. So we will build a whole new docker-compose with the different services in the same dockerfile so it will have the look of the list 5.10. From now on, there will be no need of creating any external network.

```
1  version: "3.6"
2    #docker run -p 8080:8080 cowserver:1.0
3
4  services:
5    web:
6      container_name: myapp
7      restart: always
8      environment:
9        - LORA_MONGODB_HOST=mongo
10       - LORA_MONGODB_DATABASE=lora-app
11       - ES_HOST=es01
12       - ELASTIC_URL=http://es01:9200
13       - NODE_ENV=local
14     ports:
15       - "8080:8080"
16     build: .
17     links:
```

```
18        - es01
19      depends_on:
20        - es01
21        - es02
22        - es03
23      networks:
24        - myappnetwork
25    mongo:
26      container_name: mongo
27      image: mongo
28      volumes:
29        - ./todo-mongodb-data:/var/lib/mongodb
30      ports:
31        - "27017:27017"
32
33      networks:
34        - myappnetwork
35    es01:
36      image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
37      container_name: es01
38      environment:
39        - node.name=es01
40        - cluster.name=es-docker-cluster
41        - discovery.seed_hosts=es02,es03
42        - cluster.initial_master_nodes=es01,es02,es03
43        - bootstrap.memory_lock=true
44        - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
45        - xpack.security.enabled=false
46      ulimits:
47        memlock:
48          soft: -1
49          hard: -1
50      volumes:
51        - data01:/usr/share/elasticsearch/data
52      ports:
53        - 9200:9200
54      networks:
55        - myappnetwork
56    es02:
57      image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
58      container_name: es02
59      environment:
60        - node.name=es02
61        - cluster.name=es-docker-cluster
62        - discovery.seed_hosts=es01,es03
63        - cluster.initial_master_nodes=es01,es02,es03
64        - bootstrap.memory_lock=true
65        - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
66        - xpack.security.enabled=false
```

```
67      ulimits:
68        memlock:
69          soft: -1
70          hard: -1
71      volumes:
72        - data02:/usr/share/elasticsearch/data
73      networks:
74        - myappnetwork
75    es03:
76      image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
77      container_name: es03
78      environment:
79        - node.name=es03
80        - cluster.name=es-docker-cluster
81        - discovery.seed_hosts=es01,es02
82        - cluster.initial_master_nodes=es01,es02,es03
83        - bootstrap.memory_lock=true
84        - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
85        - xpack.security.enabled=false
86      ulimits:
87        memlock:
88          soft: -1
89          hard: -1
90      volumes:
91        - data03:/usr/share/elasticsearch/data
92      networks:
93        - myappnetwork
94
95    kib01:
96      image: docker.elastic.co/kibana/kibana:7.17.1
97      container_name: kib01
98      ports:
99        - 5601:5601
100     environment:
101       ELASTICSEARCH_URL: http://es01:9200
102       ELASTICSEARCH_HOSTS: '["http://es01:9200","http://es02
      :9200","http://es03:9200"]'
103     networks:
104       - myappnetwork
105 volumes:
106   data01:
107     driver: local
108   data02:
109     driver: local
110   data03:
111     driver: local
112 networks:
113   myappnetwork:
114     driver: bridge
```

```
115     external : false
116     name: custom_network
```

Listing 5.10: functional docker-compose file for local environment,own creation

Lets make a brief description of the main changes we did to have the whole system working together at the same time in the same file. To set up the initial data gathering monitoring system we should use environment variables inside the docker compose file and we will reference to its value from inside the node server. That's why we should set up a number of environment variables in the node services. We can see that there's the ES_HOST variable for the connection to the container with the ElasticSearch service, the ELASTIC_URL, we will reference this value from inside our node server to declare the node client and fulfill the connection between the client and the search engine, and the NODE_ENV to enable the environment variables and make them accessible from inside our node server. It's remarkable the way we instantiated the node client because it was not an easy solving issue. For further info look at list 5.11

```
1 const elasticUrl = process.env.ELASTIC_URL || "http://localhost
    :9200"
2 const client = new Client({ node: elasticUrl});
```

Listing 5.11: instatiation of the ElasticSearch javascript client

This 5.11 must be donde like this because localhost between services do not see each other otherwise no connection would be possible. Notice that all the services are sharing the same network called myappnetwork.

5.5 Our final monitoring architecture design

The final proposal is a non-hierarchical, built in, decentralized monitoring system that will send a packet counting data towards a node server and a mongo database, connected with a cluster of thre nodes of ElasticSearch 6 to automatically gather all the data and process it to process all the info and get multiple visualizations in real time . It's decentralized and non hierarchical since the mesh network is sending data at the same time towards the server, there is no node gathering data from the rest, because every node is autonomous from the rest. Once the data arrives to the node server, this acts as a middleware, makes a little treatment process before the data reaches the lower layer of the system where the ElasticSearch cluster is for finally allowing us to create the visualizations and display them through the ui of kibana.

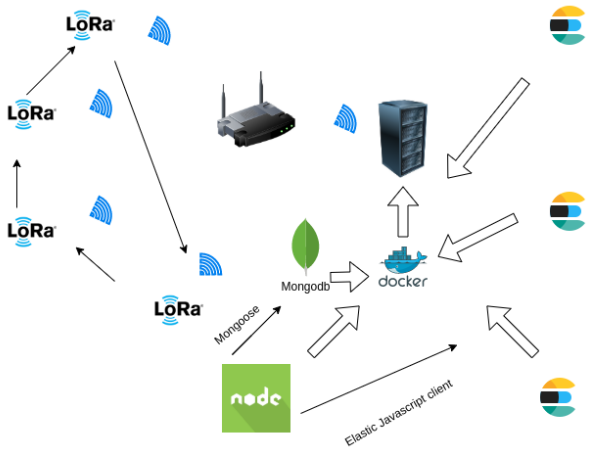


Figure 6: Monitoring System architecture,source:own creation

Chapter 6

Usage of the monitoring system

To start retrieving info from the boards, first of all we must acomplish some previous steps in our ElasticSearch server using the Kibana ui. Considering we are working in a local environment, first of all we execute 6.1

```
docker-compose up
```

Listing 6.1: docker-compose up command

6.1 Kibana setup

To prepare the Elastic environment to start the data gathering coming from the boards, first of all we should create the index in charge of storing all the values in the search engine. To do so we have to go to the hamburguer menu in the top left corner of the kibana ui and select dev tools as we point at figure 7. Then a console will show up and we should introduce the following index create script 6.2 that calls the elastic search api in order to create the index which will store all the info from the data traffic between the boards.

```
PUT /monitorization3
{

  "mappings" : {
      "properties" : {
        "broadcast" : {
          "type" : "integer"
        },
        "datapackme" : {
          "type" : "integer"
        },
        "dstinyunreach" : {
          "type" : "integer"
```

```
14          },
15          "fwdpackets" : {
16            "type" : "integer"
17          },
18          "iamvia" : {
19            "type" : "integer"
20          },
21          "localaddress" : {
22            "type" : "keyword"
23          },
24          "notforme" : {
25            "type" : "integer"
26          },
27          "packetsforme" : {
28            "type" : "integer"
29          },
30          "rechellopackets" : {
31            "type" : "integer"
32          },
33          "recpackets" : {
34            "type" : "integer"
35          },
36          "sendhellopackets" : {
37            "type" : "integer"
38          },
39          "sendpackets" : {
40            "type" : "integer"
41          },
42          "totalreceived" : {
43            "type" : "integer"
44          },
45          "senddatapackets": {
46            "type": "integer"
47          },
48          "timestamp":{
49            "type": "date"
50            }
51          }
52        }
53      }
```

Listing 6.2: index creation api call script,source:own creation

The next two steps are going to analytics, in the same hamburguer menu and then select manage in the righ top corner. Then look at the left side and there's an option called index patterns, then choose create index pattern and select monitorization3 as the index pattern to be created. From now on the user is able to build any dashboard with any of the available plots.

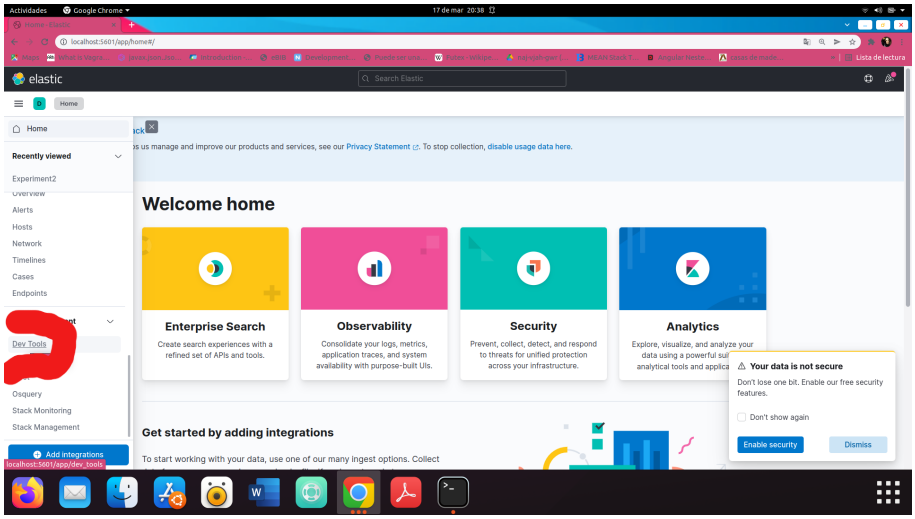


Figure 7: Dev tools menu location

6.2 Flashing boards

You should think about what main code you want to flash on your LoRa boards. For that task you can do it one by one using the Platformio IDE pressing the arrow at the bottom of the visual code environment. The other way is executing the script we show at list number A.1 at the end of this document while the boards are plugged in to the usb ports of your pc.

6.3 Plotting Dashboards

The dashboards option is one of the first options on the hamburguer menu at the left side of the screen. Then you should choose create a new visualization. Then there are a lot types of plots as an option, a Kibana Query Language console where the user can execute queries to filter data to visualize, and on the right side of the dashboards interface there are the parameters of the plot. The user shall choose the fields of the X axis, usually the timestamp, and the Y axis with a serie of operations such as median, maximum, count and many more. And there is a break by that it would match with the group by of a standard SQL query. Besides this any kind of filters can be applied to the query in KQL(Kibana Query Language) to obtain the right visualization of the data. In the top bar there is short menu where there is the time interval the user wants to display and set up the refreshing period to display new incoming data. There are as many visualizations as someone would desire, and these can be stored into the same dashboard to check the status

of the network wathching several parameters at the same time. The visualizations can be edited at any moment. Once the desired dashboard configuration is set up,save the dashboard and it will be available at any time.

6.4 First Experiments

To test all the set up, we planned an initial experiment. We grabbed four LoRa boards with the following main program showed in list 6.3

```
1
2  if (radio->getLocalAddress() == 0xC4F0&& dataCounter1 <= 60) {
3          Log.trace(F("Send packet %d" CR), dataCounter1);
4          helloPacket->counter = dataCounter1++;
5
6          //Create packet and send it.
7          radio->createPacketAndSend(0xD39C, helloPacket, 1);
8      }
9      if(radio -> getLocalAddress() == 0xD39C&& dataCounter2 <= 60){
10         Log.trace(F("Send packet %d"CR),dataCounter2);
11         helloPacket->counter = dataCounter2++;
12         radio->createPacketAndSend(0x3A0C,helloPacket,1);
13     }
14     if(radio -> getLocalAddress() == 0x3A0C&& dataCounter3 <= 60){
15         Log.trace(F("Send packet %d"CR),dataCounter3);
16         helloPacket->counter = dataCounter3++;
17         radio->createPacketAndSend(0xB1A4,helloPacket,1);
18         }
19
20         if(radio -> getLocalAddress() == 0xB1A4 && dataCounter4 <=
       60){
21         Log.trace(F("Send packet %d"CR),dataCounter4);
22         helloPacket->counter = dataCounter4++;
23         radio->createPacketAndSend(0xC4F0,helloPacket,1);
24     }
25
26     //Wait 5 seconds to send the next packet
27     vTaskDelay(5000 / portTICK_PERIOD_MS);
28     Log.trace(F("Another while loop"CR));
```

Listing 6.3: main.cpp, source:own creation

Initial conditions:

- Number of boards involved: 4
- Duration of the experiment: 5 minuts
- Periodicity of data packets: 5 seconds

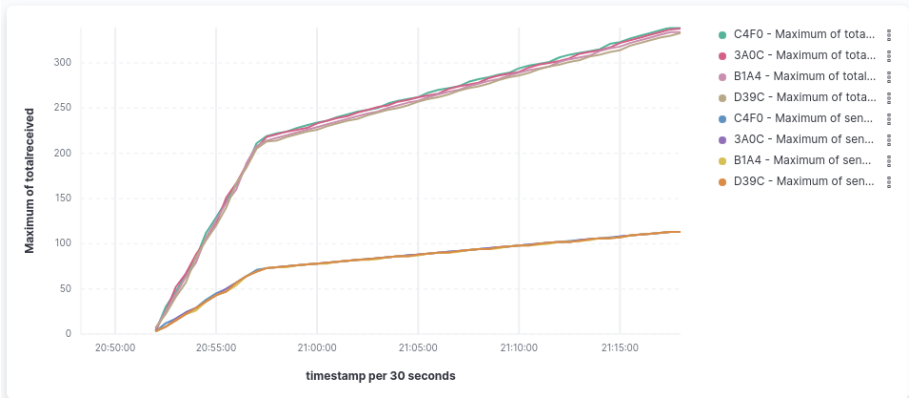


Figure 8: Relation between received and sent packets in four boards,source:kibana

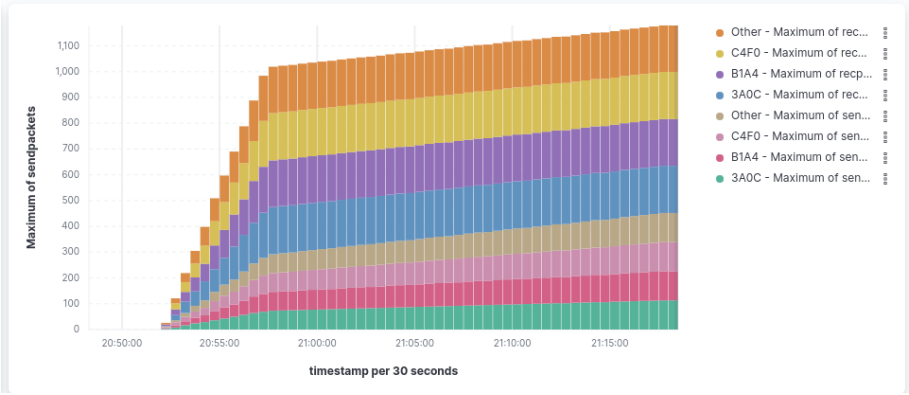


Figure 9: Relation between received and sent packets in four boards in a bar diagram, source: kibana

- Periodicity of hello packets: 30 seconds
- Expected behaviour: in/out packet traffic moving in a linear way and see how the number of send received packets match and there is no packet loss.

As we see it only consists of four boards sending a counter to a different board each one every 5 seconds during five minuts. We will see the behaviour of it with a couple of plots.
We can see in fig 9 the packets being received by the four boards, and the line below are the packets being send. After five minuts the line slope decreases because there are no more data packets to send . We have another graphic only for show the possibilities of this system, the one in figure 9
In this last graphic showed in figure 10 we can see the relation in the four boards. We can see clearly the traffic between the four boards, we can see even how the

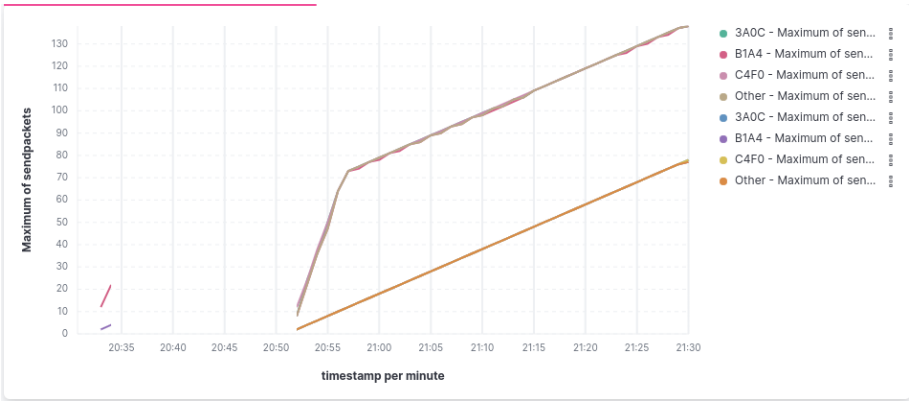


Figure 10: Relation between sent packets and in four boards and sent hello packets, source: kibana

new boards are being incorporated to the network in the first bars. Notice that the line below are the packets being send through the network, this matches with the reality expected since the boards are communicating with the remaining boards, so the received packets will be always a bigger number than the sent, since the they are receiving packets from all the rest aproximately three times more packets since one board is receiving packets from the remaining three boards. We don't see significant packet losses, however we could place the mouse over the lines and kibana will show us the exact number of received packets and we can see exactly whether a packet has been lost or not. We see the packet traffic is homogeneous along the whole network as well, all the lines are one over the other, so we see no difference among the nodes. In the third graphic we see the weight of the hello packets in the total of sent packets. We clearly see after five minutes, the slope of both lines is the same, only the hello packets are being transmitted what makes a lot of sense.

6.4.1 Longer-Term monitoring experiments

Now as a proof of a long term monitoring experiment
Initial conditions:

- Number of boards involved: 4
- Duration of the experiment: 24 hours
- Periodicity of data packets: 30 seconds
- Periodicity of hello packets: 5 minuts

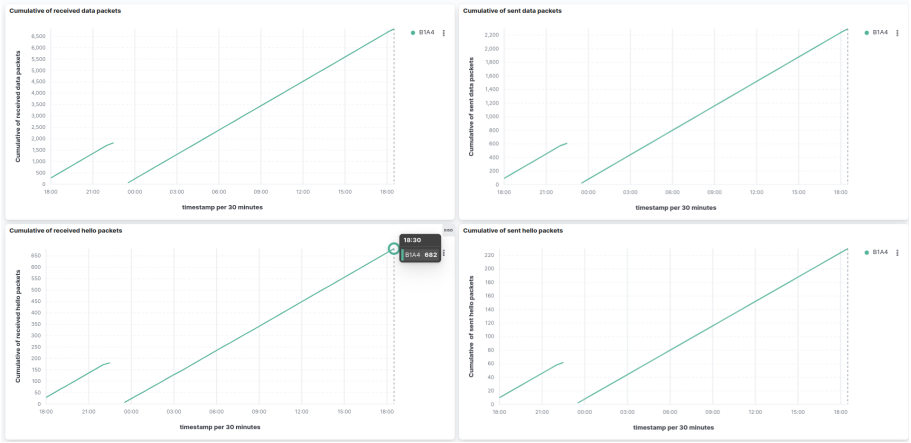


Figure 11: Relation between sent packets and received packets in board B1A4, including routing and data packets, source: kibana

- Expected behaviour: in/out packet traffic moving in a linear way and see how the number of send received packets match and there is no packet loss.We will also check the traffic of the main types of packets and see wether the traffic is homogenous among all the nodes and see other kinds of visualizations availables in kibana

The main program will be the same as the previous experiment. One board will transmit a data packet to the other in circle, until the last board that will send a packet to the first, and the network will remain like this during the whole experiment. This time we will try to obtain a different view of the traffic of the data trying to spot the relation between the sent hello packets and received hello packets in one particular board, and we will do the same with the data packets. For that task we have generated the dashboard in fig 11.
We can see that the time scope of this plot is the first 18 hours. We can see clearly the time when we pressed the reset button on the boards, and from that moment we have an increasing line along time. Notice the values and the relation between the sent packets and the received packets, in both cases the sent packets are a third from the received ones which matches with what we expected since we have three boards trasnsmitting. Since the data packets are being sent more often than the routing ones, the values of the data plot are a lot bigger than the values of the hello packets what makes a lot of sense. From that moment the slope of the line remains constant. We want to show one more thing about this kind of plot because this take it's difficult to appraise. That is why we will take a closer look to a smaller time interval so we select with the mouse the interval we want to look at and we obtain the image in figure 12.
In this last we do see how the period of the routing packet sending routine is far

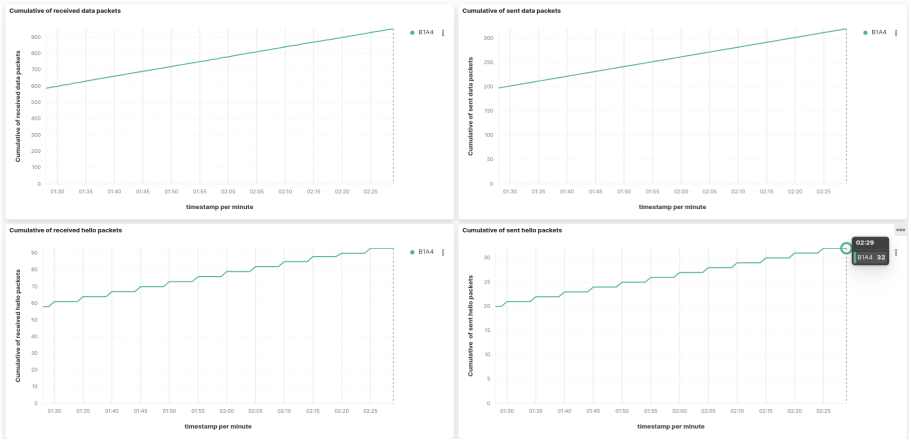


Figure 12: Relation between sent packets and received packets in board B1A4, including routing and data packets(ZOOM IN), source: kibana

way bigger than data sending routine, that's why we see this stairway disposition of the line along the time corresponding every five minutes to one step higher but we weren't unable to see it from a plot with such a big interval in the previous graphic in fig 11. Besides this tipical line plot along the time we would like to see if the data and routing packets flow in a homogeneus way along all the nodes in the mesh network, so we will find out what more visualizations options do we have in kibana . So lets other ways of displaying data.
From the image in figure 13 we see that the traffic is homogeneous along the whole network. All the nodes are one hop of distance so It matches with what we expect from this network configuration. We will do a last visualization proposal. It's important to see wether has been a packet loss issue or not so we should figure out what would be a good way of having that info in a quick outlook, or at least see if a deep fall in the performance of the network is happening . For that milestone we could plot a graphic like the coming one in figure 14.
We should check that in every node the received data packets are three times bigger than the sent ones. To see if there has been a single packet loss issue we could place the the mouse at any of these bars and check the exact value and do the maths, but from this point of view we notice that all the green bars are three times the blue ones, so we can say that our network has quite good health. Of course there are plenty of ways of obtaining many kinds of plots and charts and Kibana has an unending set of options and features but our goal in this Bachelor Thesis is only give a little taste of some of them.

Figure 13: Pie chart representing sent packets and received packets along all the nodes, including routing and data packets, source: kibana

Figure 14: Vertical stacked bar chart representing sent packets and received packets along all the nodes, including routing and data packets, source: kibana

Chapter 7

Deploying to a public IP

To deploy the monitoring server to a public IP, first of all it would be suitable to enable the security of the Elastic Stack. But first of all we thought that we should get used to the UPC server environment.

7.1 First Test Deployment

We had forseen that a moment like this was about to come, so we have been developing the whole project in docker containers to once arrived this moment, the deployment to a public IP can be as easy as possible. It was not that easy though It took only a several steps :

1. Give my public key to the server system administrator in order to get access granted via ssh
2. Install docker with the into the clean server with the next commands in list 7.1

```
1     sudo apt update
2     sudo apt install apt-transport-https ca-certificates curl
      software-properties-common
3     curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
      sudo apt-key add -
4     sudo add-apt-repository "deb [arch=amd64] https://
    download.docker.com/linux/ubuntu focal stable"
5     apt-cache policy docker-ce
6     sudo apt install docker-ce
7
```

Listing 7.1: Docker installation commands,source:[8]

3. Installing and docker-compose with the following commands in list 7.2

```
1     mkdir -p ~/.docker/cli-plugins/
2 curl -SL https://github.com/docker/compose/releases/download/
     v2.2.3/docker-compose-linux-x86_64 -o ~/.docker/cli-
     plugins/docker-compose
3 chmod +x ~/.docker/cli-plugins/docker-compose
4 docker compose version
5
```

Listing 7.2: Intalling docker-compose commands,source: [7]

4. Install git
5. git clone https://github.com/pellax/micowlocalyzer.git to clone repository into your directory
6. git fetch origin to fetch all remote branches
7. git checkout -b singlenode origin/singlenode
8. git pull
9. Perform step in section A.0.7
10. Execute docker-compose up
11. Wait for the message kibana is degraded in the log console, now your kibana set up is ready.
12. In the Kibana application, enter to the dev tools and create the index with the script in list 6.2

Now the system is deployed to a public IP and for a quick test would be enough, but since this is accesible to anyone this can not be considered a stable deployment so this needs to be properly securized in order to restrict access to unwanted guests.

7.2 Securized deployment

At the time we wanted to acomplish a securized version of our containerized Elastic Stack we faced several previous questions.

7.2.1 First steps

The initial dockerized version was working with the elastic 7.17.1 but in the way of developing the server, the 8 release came out . Some security options remained the same, but there were some new features and some other were deprecated. We found ourselves in the tough decision of remaining in the old version or make a breakthrough into the new one. The new version had the following security options to be enabled. Lets take a quick overview of it

7.2.2 ELK stack security features

Security needs vary depending on whether the user is developing locally on his laptop or securing all communications in a production environment. Regardless of where the deploying the Elastic Stack ("ELK") is being carried out, running a secure cluster is incredibly important to protect the data. That's why security is enabled and configured by default in Elasticsearch 8.0 and later. In our initial version of ElasticSearch, the number 7.17.1, security was disabled by default. But in a week period of time the version 8 release came out and this specific feature was disabled so we had to add the following parameter in list 7.3 to our docker compose file to disable the security explicitly for our local deployment.
If we configure security manually before starting our Elasticsearch nodes, the auto-configuration process will respect the security configuration. We can adjust your TLS configuration at any time, as well as updating node certificates[12].

```
1 xpack.security.enabled=false
```

Listing 7.3: docker-compose security parameter

There are two main options(fig 15) to configure security in the transport layer on the Elastic Stack These is a brief description of both of them :

- **Basic security (ElasticSearch + Kibana)** This scenario configures TLS for communication between nodes. This security layer requires that nodes verify security certificates, which prevents unauthorized nodes from joining our ElasticSearch cluster.

  The external HTTP traffic between ElasticSearch and Kibana won't be encrypted, but internode communication will be secured.

- **Basic security plus secured HTTPS traffic (Elastic Stack)** This scenario builds on the one for basic security and secures all HTTP traffic with TLS. In addition to configuring TLS on the transport interface of the ElasticSearch cluster, the user configures TLS on the HTTP interface for both ElasticSearch and Kibana.

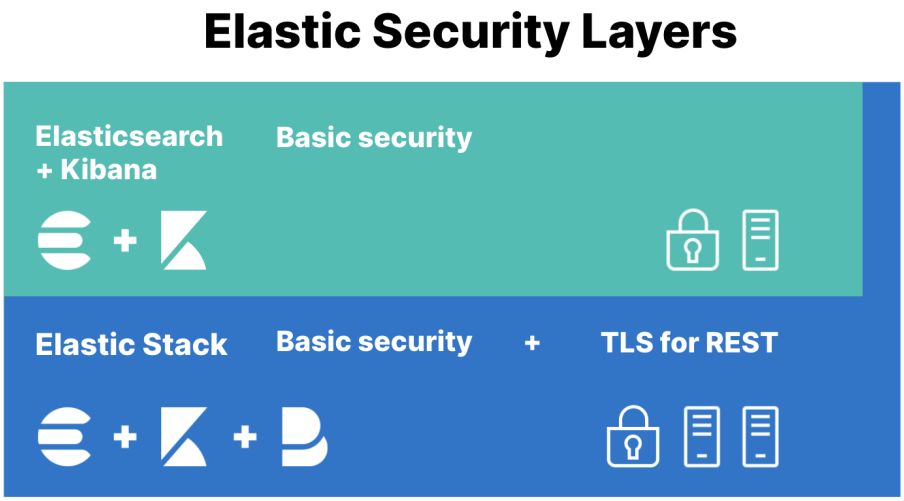


Figure 15: Elastic security overview,source:Elastic site

7.2.3 Securizing the ELK Stack

For the basic security we have to follow this steps so it's something we have to do anyway.

1. Generate the certificate authority We can add as many nodes as we want in a cluster but they must be able to communicate with each other. The communication between nodes in a cluster is handled by the transport module. To secure our cluster, we must ensure that internode communications are encrypted and verified, which is achieved with mutual TLS.

   In a secured cluster, ElasticSearch nodes use certificates to identify themselves when communicating with other nodes.

   The cluster must validate the authenticity of these certificates. The recommended approach is to trust a specific certificate authority (CA). When nodes are added to our cluster they must use a certificate signed by the same CA.[12]

   For the transport layer, they recommend using a separate, dedicated CA instead of an existing, possibly shared CA so that node membership is tightly controlled. For this first deployment though we will use self signed certificates.

2. Encrypt internode communications with TLS

   The transport networking layer is used for internal communication between nodes in a cluster. When security features are enabled, we must use TLS to ensure that communication between the nodes is encrypted.

> Now that we've generated a certificate authority and certificates, we'll update our cluster to use these files. [12]

That's why we must modify our dockerfile. First of all we should generate all the certificates that will be using to encrypt the internode communication . That's why we should implement another docker compose file to generate all the certificates. In this case before generating the certificates, we will define the instances of our elastic system in a file called instances.yml as we show in list 7.4

```
instances:
  - name: es01
    dns:
      - es01
      - localhost
    ip:
      - 127.0.0.1

  - name: es02
    dns:
      - es02
      - localhost
    ip:
      - 127.0.0.1

  - name: es03
    dns:
      - es03
      - localhost
    ip:
      - 127.0.0.1
  - name: kib01
    dns:
      - kib01
      - localhost
```

Listing 7.4: instances.yml, source : [2]

Once we have defined the instances, wa are able to execute the following docker-compose file in list 7.5 to in fact, create the certificates for each one of these instances.

```
version: '2.2'

services:
  create_certs:
    container_name: create_certs
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.1
    command: >
      bash -c '
```

```
9         if [[ ! -f /certs/bundle.zip ]]; then
10          bin/elasticsearch-certutil cert --silent --pem --in
    config/certificates/instances.yml -out /certs/bundle.zip;
11          unzip /certs/bundle.zip -d /certs;
12        fi;
13        chown -R 1000:0 /certs
14      '
15    user: "0"
16    working_dir: /usr/share/elasticsearch
17    volumes: ['certs:/certs', '.:/usr/share/elasticsearch/config/
    certificates']
18
19 volumes: {"certs"}
```

Listing 7.5: create-certs.yml,source :[2]

After executing this container, we must set up some environment variables with the following values, so the .env file located in the same folder as the docker-compose file, should look like list 7.2.3:

```
1 COMPOSE_PROJECT_NAME=es
2 CERTS_DIR=/usr/share/elasticsearch/config/certificates
3 VERSION=7.17.1
```

So finally we will modify the docker-compose file adding the proper parameters. We show the result in the list 7.6

```
1  version: "2.2"
2    #docker run -p 8080:8080 cowserver:1.0
3
4  services:
5    web:
6      container_name: myapp
7      restart: always
8      environment:
9        - LORA_MONGODB_HOST=mongo
10       - LORA_MONGODB_DATABASE=lora-app
11       - ES_HOST=es01
12       - ELASTIC_URL=http://es01:9200
13       - NODE_ENV=local
14     ports:
15       - "8080:8080"
16     build: .
17     links:
18       - es01
19     depends_on:
20       - es01
21       - es02
22       - es03
23     networks:
24       - myappnetwork
```

```
25   mongo:
26     container_name: mongo
27     image: mongo
28     volumes:
29       - ./todo-mongodb-data:/var/lib/mongodb
30     ports:
31       - "27017:27017"
32
33     networks:
34       - myappnetwork
35   es01:
36     image: docker.elastic.co/elasticsearch/elasticsearch:${VERSION
       }
37     container_name: es01
38     environment:
39       - node.name=es01
40       - cluster.name=es-docker-cluster
41       - discovery.seed_hosts=es02,es03
42       - cluster.initial_master_nodes=es01,es02,es03
43       - bootstrap.memory_lock=true
44       - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
45       - xpack.license.self_generated.type=trial
46       - xpack.security.enabled=true
47       - xpack.security.http.ssl.enabled=true
48       - xpack.security.http.ssl.key=$CERTS_DIR/es01/es01.key
49       - xpack.security.http.ssl.certificate_authorities=$CERTS_DIR
       /ca/ca.crt
50       - xpack.security.http.ssl.certificate=$CERTS_DIR/es01/es01.
       crt
51       - xpack.security.transport.ssl.enabled=true
52       - xpack.security.transport.ssl.verification_mode=certificate
53       - xpack.security.transport.ssl.certificate_authorities=
       $CERTS_DIR/ca/ca.crt
54       - xpack.security.transport.ssl.certificate=$CERTS_DIR/es01/
       es01.crt
55       - xpack.security.transport.ssl.key=$CERTS_DIR/es01/es01.key
56     ulimits:
57       memlock:
58         soft: -1
59         hard: -1
60     volumes:
61       - data01:/usr/share/elasticsearch/data
62       - certs:$CERTS_DIR
63     ports:
64       - 9200:9200
65     networks:
66       - myappnetwork
67     healthcheck:
68       test: curl --cacert $CERTS_DIR/ca/ca.crt -s https://
```

```
    localhost:9200 >/dev/null; if [[ $$? == 52 ]]; then echo 0;
    else echo 1; fi
69      interval: 30s
70      timeout: 10s
71      retries: 5
72
73  es02:
74    image: docker.elastic.co/elasticsearch/elasticsearch:${VERSION
    }
75    container_name: es02
76    environment:
77      - node.name=es02
78      - cluster.name=es-docker-cluster
79      - discovery.seed_hosts=es01,es03
80      - cluster.initial_master_nodes=es01,es02,es03
81      - bootstrap.memory_lock=true
82      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
83      - xpack.license.self_generated.type=trial
84      - xpack.security.enabled=true
85      - xpack.security.http.ssl.enabled=true
86      - xpack.security.http.ssl.key=$CERTS_DIR/es02/es02.key
87      - xpack.security.http.ssl.certificate_authorities=$CERTS_DIR
    /ca/ca.crt
88      - xpack.security.http.ssl.certificate=$CERTS_DIR/es02/es02.
    crt
89      - xpack.security.transport.ssl.enabled=true
90      - xpack.security.transport.ssl.verification_mode=certificate
91      - xpack.security.transport.ssl.certificate_authorities=
    $CERTS_DIR/ca/ca.crt
92      - xpack.security.transport.ssl.certificate=$CERTS_DIR/es02/
    es02.crt
93      - xpack.security.transport.ssl.key=$CERTS_DIR/es02/es02.key
94    ulimits:
95      memlock:
96        soft: -1
97        hard: -1
98    volumes:
99      - data02:/usr/share/elasticsearch/data
100     - certs:$CERTS_DIR
101   networks:
102     - myappnetwork
103 es03:
104   image: docker.elastic.co/elasticsearch/elasticsearch:${VERSION
    }
105   container_name: es03
106   environment:
107     - node.name=es03
108     - cluster.name=es-docker-cluster
109     - discovery.seed_hosts=es01,es02
```

```
110       - cluster.initial_master_nodes=es01,es02,es03
111       - bootstrap.memory_lock=true
112       - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
113       - xpack.license.self_generated.type=trial
114       - xpack.security.enabled=true
115       - xpack.security.http.ssl.enabled=true
116       - xpack.security.http.ssl.key=$CERTS_DIR/es03/es03.key
117       - xpack.security.http.ssl.certificate_authorities=$CERTS_DIR
      /ca/ca.crt
118       - xpack.security.http.ssl.certificate=$CERTS_DIR/es03/es03.
      crt
119       - xpack.security.transport.ssl.enabled=true
120       - xpack.security.transport.ssl.verification_mode=certificate
121       - xpack.security.transport.ssl.certificate_authorities=
      $CERTS_DIR/ca/ca.crt
122       - xpack.security.transport.ssl.certificate=$CERTS_DIR/es03/
      es03.crt
123       - xpack.security.transport.ssl.key=$CERTS_DIR/es03/es03.key
124     ulimits:
125       memlock:
126         soft: -1
127         hard: -1
128     volumes:
129       - data03:/usr/share/elasticsearch/data
130       - certs:$CERTS_DIR
131     networks:
132       - myappnetwork
133   kib01:
134     image: docker.elastic.co/kibana/kibana:${VERSION}
135     container_name: kib01
136     depends_on: {"es01": {"condition": "service_healthy"}}
137     ports:
138       - 5601:5601
139     environment:
140       SERVERNAME: localhost
141       ELASTICSEARCH_URL: https://es01:9200
142       ELASTICSEARCH_HOSTS: https://es01:9200
143       ELASTICSEARCH_USERNAME: kibana_system
144       ELASTICSEARCH_PASSWORD: YU8K7Jrwuk9aVrGczzDe
145       ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES: $CERTS_DIR/ca/ca.
      crt
146       SERVER_SSL_ENABLED: "true"
147       SERVER_SSL_KEY: $CERTS_DIR/kib01/kib01.key
148       SERVER_SSL_CERTIFICATE: $CERTS_DIR/kib01/kib01.crt
149     volumes:
150       - certs:$CERTS_DIR
151     networks:
152       - myappnetwork
153 volumes:
```

```
154   data01:
155     driver: local
156   data02:
157     driver: local
158   data03:
159     driver: local
160   certs:
161     driver: local
162 networks:
163   myappnetwork:
164     driver: bridge
165     external : false
166     name: custom_network
```

Listing 7.6: modified docker-compose file, source: own creation

From the lines above we see that we have added several parameters to each node. In the previous docker compose file we have generated all the certificate, so we had to add an extra volume in each node to store these certificates.
We will make a brief explanation of the changes we have introduced in the docker file to have the whole system working with the security enabled.

1. Add the cluster-name setting and enter a name for your cluster

```
1     - cluster.name=elasticsearch-cluster
```

2. We should add node name in every node of elasticsearch.

```
1     node.name=elasticsearch01
```

3. We must enable security in every node

```
1     - xpack.security.enabled=true
```

4. We must set certificated mode to required

```
1     - xpack.security.transport.ssl.verification_mode:
      certificate
2
```

5. We define certificate key location

```
1   - xpack.security.http.ssl.key=$CERTS_DIR/es03/es03.key
2
```

6. We define certificate location

```
1   - xpack.security.http.ssl.key=$CERTS_DIR/es03/es03.crt
2
```

7. We define as well certificate authorities location

```
1   - xpack.security.http.ssl.certificate_authorities=
      $CERTS_DIR/ca/ca.crt
2
```

8. We define transport layer certificate key location

```
1 - xpack.security.transport.ssl.key=$CERTS_DIR/es03/es03.key
2
```

9. We define transport layer certificate location

```
1   - xpack.security.transport.ssl.certificate=$CERTS_DIR/es03/
      es03.crt
2
```

10. We define as well transport layer certificate authorities location

```
1   - xpack.security.transport.ssl.certificate_authorities=
      $CERTS_DIR/ca/ca.crt
2
```

11. Allow self generated certificates

```
1       - xpack.license.self_generated.type=trial
2
3
```

Concerning to the Kibana service we should add some parameters as well to the kibana service

1. Enable security

```
1 - XPACK_SECURITY_ENABLED=true
2
```

2. Set the path to the location of the of the SSL certificate authorities

```
1 -ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES: $CERTS_DIR/ca/ca.
     crt
2
```

3. Enable inbound connections

```
1 - SERVER_SSL_ENABLED=true
2
```

4. Give access to the certificate and the private key and certificate

```
  SERVER_SSL_KEY: $CERTS_DIR/kib01/kib01.key
  SERVER_SSL_CERTIFICATE: $CERTS_DIR/kib01/kib01.crt
```

5. Set elastic credentials to set up the communication between kibana and elastic engine consisting in username and generated password

```
      ELASTICSEARCH_USERNAME: kibana_system
      ELASTICSEARCH_PASSWORD: YU8K7Jrwuk9aVrGczzDe

```

The last thing we need is to change all the addresses of the hosts to https instead of http. The other main change we have to do is in the server code, in the way we instantiate the javascript elastic client in the constructor. Now we will do it as we show in list 7.7

```
module.exports.getClient = function(){
const elasticUrl = process.env.ELASTIC_URL || "https://localhost
    :9200"
const client = new Client({ node: elasticUrl,auth:{username: '
    elastic',password:'PleaseChangeMe'
}
});
```

Listing 7.7: modded client instantiation,source:own creation

7.2.4 Loading the whole system

Now we have all the containers ready, the instances, the ELK container configurated, the certificates creator container written we have to put it all together and get it working. We have to carefully follow this steps [2]

1. Ensure we have at least 4 GiB of memory allocated.
2. Execute the following in list 7.8 docker-compose up command to create the certificates

```
    docker-compose -f create-certs.yml run --rm create\_certs

```

Listing 7.8: Certificates creation,source:[2]

3. Lift the container with your ELK stack with a docker-compose up -d
4. Execute the following commands in list 7.9 that will execute the instructions inside the container to generate the passwords to our ELK stack. We have to keep these passwords somewhere safe because we will use them to connect

the services between them. The kibana system will remain as the user for the connect the kibana to the elastic engine and the elastic user will be for loggin in into the kibana ui and submit requests to the Elastic Stack.

```
1     docker exec es01 /bin/bash -c "bin/elasticsearch-setup-
      passwords \
2 auto --batch --url https://es01:9200"
3
```

Listing 7.9: Password creation command,source :[2]

5. Now we can set the ELASTIC_PASSWORD in the kibana service corresponding to the kibana_system user generated password

6. Now we have to restart our docker container to reload the security configuration so we have to execute the corresponding commands in list 6

```
1     docker-compose stop
2     docker-compose up -d
3
```

7. Now our cluster will be available at the address https://ourIP:5601

Trying to establish connection between the node client and the elastic server as it is we saw that the system was complaining for trying to send plain traffic through an https port . So we must set up a place to store the certificate in the node server as well. But this will be a task for the future work, we will talk about it in the conclusions.

Chapter 8

Conclusions

We will divide our conclusions in two parts, the technical and the personal.

8.1 Technical conclusions

We wanted to make an statistical study about the performance of LoRaMesher. Instead of this we ended building a whole monitoring system that will allow us to do experiments in an easier and faster way saving a lot of time for future researchs and allowing to monitorize LoRaMesher in different scenarios. The system seems to be minimum invasive and seems to work well with four boards and the performance doesn't look threatened while working with four boards. To sum up we did a test deploy to a public IP and we left a future securization almost done to prevent unwanted users to access to our system. We need to perform further experiments as we point in section 8.3 to calculate the differents kinds of overhead, as well as execute some more experiments to quantify exactly the behaviour of the LoRaMesher combined with our monitoring system to ensure that there is no fall on the protocol performance degree.As we will see in section 8.3 we have already done some researchs about what things we should consider at the time we will be doing the calculus of the post request, but it would be suitable to know the overhead of the server response as well.However our system is the first step towards a wireless real time monitoring system and it will be extremely useful to monitor mesh networks, concretely this LoRaMesher protocol.

8.2 Personal conclusions about the work and other comments

At the beginning of the work I was given this initial library implementing a mesh network protocol designed by UPC Phd. Roger Pueyo Centelles . The code was a very initial implementation of this protocol, I was supposed to do some initial experiments, fix the protocol and plot some final data extracted from the boards. Due to several issues I couldn't meet the deadlines in December and the new year came in. Thats when my thesis suffered a great deviation.First of all I had a new version of the LoRaMesher that made my way easier.To sum up I saw myself trying to extract data from the mongo database and I found it quite messy. That's why I decided to take advantatge of the elastic javascript client that would allow us to take our already build node server and make a data flow towards the search engine automatically, and that was finally what this thesis was about. We tested it in local and It worked, and then I was asked to do a public deployment. I did a test deployment and I started a research about how would it be a full securized deployment which I was unable to end but luckily I've been hired by the AC department and with a little more time I'll be in position of doing an stable and longing full usable deployment to a server in the AC department.To measure the goodness of this new mesh networks monitoring system there are some things that still need to be done. We will talk about them in the coming section 8.3

8.3 Future work

8.3.1 Monitoring system stress experiments

To see how much intrusive is our monitoring system we should perform some more experiments with some more boards and look harder whether there is a packet loss or not. There can be several sources of packet loss issues, one can be the protocol working process itself, and the other could be our monitoring sytem. We should evaluate better our monitoring system in different scenarios. It reminds a little bit to the Schrodinger paradox, we need a monitoring system to know if there is a packet loss, but our monitoring system can cause a packet loss issue as well and It's difficult to say whose its fault. Of course there should be other ways of less intrusive traffic monitoring through the logs, but you need the boards to be plugged in to a computer and this is not the subject of this thesis.

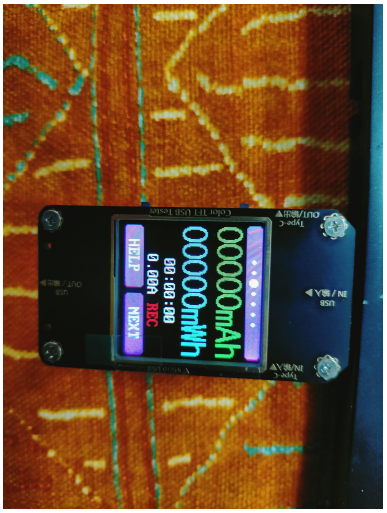


Figure 16: Digital usb multimeter

8.3.2 Experiments about the best fit in the priority task scale

We have decided to set the task priority to 2 but this is not apodictically the best choice. We should try other priorities and do some experiments and check which one is in fact the best priority value to get the best performance.

8.3.3 Measuring monitoring systems overheads

Adding one the taks to an already built protocol implies several overheads at many levels. We should estimate as close as possible the following overheads.

8.3.3.1 Power Consumption overheads

We have done already some experiments with the little USB power meter in figure 16
We face several problems trying to use this device. The main one is that the power consumption doesn't remains stable so we can't have a fix value about the consumption of the board. This multimeter allows data transmission through an Android device or a PC via Blueetooth and a application. We tried the app for android but we couldn't establish data flow between botth devices. We were about to try it with the pc and see wether we were able to obtain some kind of plot an do a comparation between the plot with the additional task and without it and see if there is any significative differences. The other option was taking out all the

tasks from the scheduler and leave our post sending task alone in an infinite loop and see wether we can get a stable value from it .

8.3.3.2 Post Requests Overheads

We should see which is the exact size of every post request between the boards and the node server. And we could do the same with the requests from the Javascript client to the elastic search engine. Luckily there are some tools that would allow us to do this easily . The classic way of doing this would be storing a trace using the well known tool TCPdump. Afterwards we should open the trace using wireshark and calculate the packet size and the headers of an http request. But first of all we must know what we understand by post request overhead.

What is the overhead The post request is divided in several parts. On one side we have the payload which is data it self being sent through the request. The request has a header which also occupies a fixed size. In this case we have TCP header which is bigger than UDP for example. That's why one of the things that could be improved to reduce the size of the transfered data,which means that UDP has smaller overhead. We understand by overhead, all the additional data that we need to send in order to transmit our payload. TCP has bigger overhead, among other reasons, TCP has a mechanism to ensure that all the data has been received, has a number of sequence and other things that UDP doesn't . In the coming lines we will talk about some previous work that we have done to face the future calculus of the overhead.

Obtaining the trace First of all,we will need to save the trace into the disk to be analyzed afterwards with wireshark. Execute a command like the one in list 8.1 in the server where we have deployed our node server.

```
1 tcpdump -i eth0 port 8080 -nw mycapture.pcap
```

Listing 8.1: tcpdump command for extracting the trace,source:own creation

Analyzing the trace Once we have the trace generated we should analyze it and see what kind of info we can obtain from. From figure 17 and figure 18 we can see some packet sizes. Among them we have :

- Ip header size : 20 bytes
- IP total packet size: 125 bytes
- TCP header size: 20 bytes

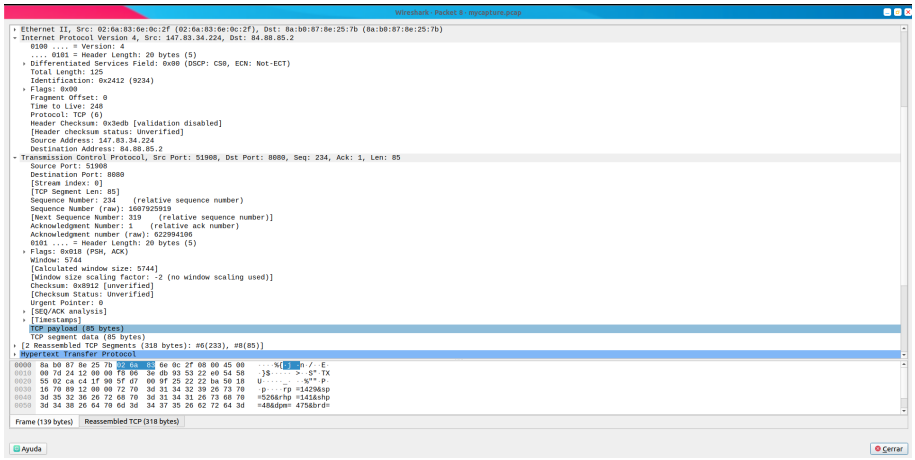


Figure 17: LoRaMesher tcpdump trace, source: wireshark gui

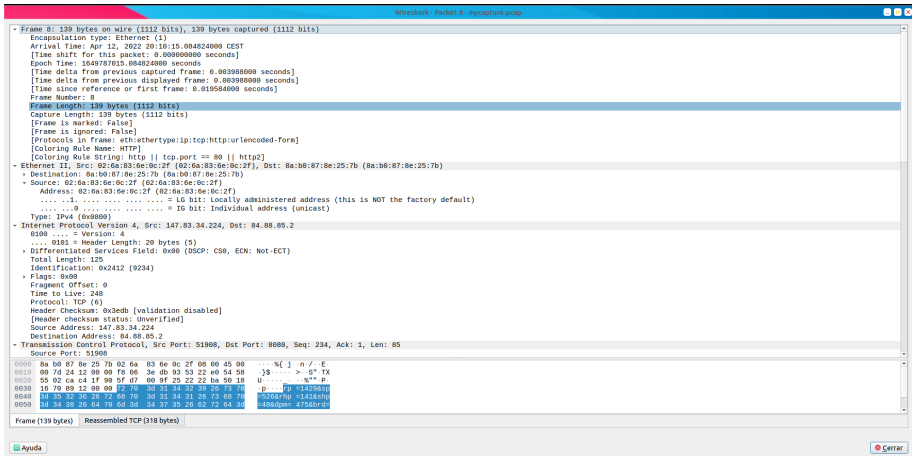


Figure 18: LoRaMesher tcpdump trace, source: wireshark gui

- TCP payload : 85 bytes
- Total frame length: 139 bytes

To calculate the theoretical overhead there are some previous concepts we will have to consider.

- MSS: The Maximum Segment Size refers to size of the largest segment that local host accepts within a single packet. It denotes largest amount of data that host can accept in single TCP segment [13]. This value is the result of the agreement between the sender and the receiver. During a TCP connection the sender may reduce the value of the MSS according to the receiver requirements. It's the way that TCP has to set a limit on the size of the

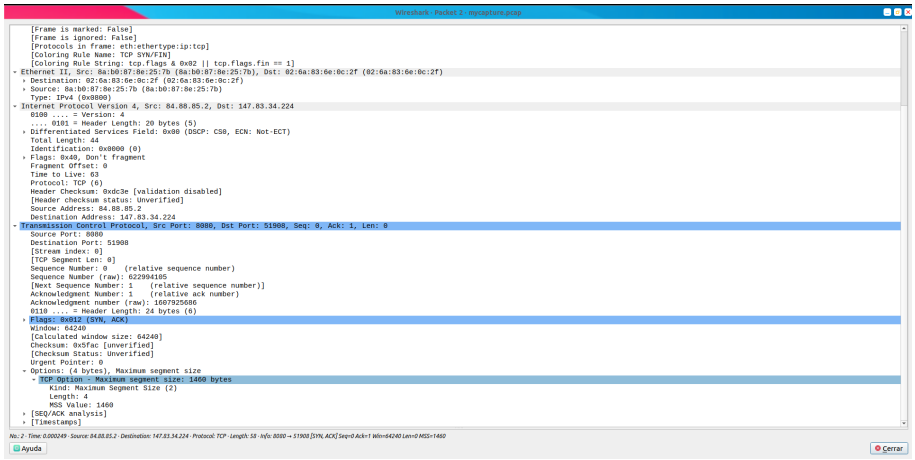


Figure 19: LoRaMesher tcpdump trace, source: wireshark gui

received packets. We can see the value of this agreeement for this tcp session is set to 1460 from figure 19 which is a very common value.

- MTU: The Maximum Transmission Unit is the largest packet size that can be transmitted in a single entity in a network connection [14]. By default this value is set to 1500 B.

We should keep in mind as well the fig 20 with the structure of a tcp packet.
All the theory behind the calculus of the overhead is quite easy. We have the following theoretical overhead in every Ethernet packet is :

$$Overhead = 20B(IPHeader) + 20B(TCPHeader) + 14B(EthernetHeader) + 4B(FCS) + 12B(Interframegap) + 8B(Preamble) = 78$$

The payload it's supposed to be the MTU- headers size that gives us as a result the MSS value, 1460 bytes. However if we look at the figures 17 and 18 we see that the tcp payload is much lower than the MSS, as well as the size of the whole packet is smaller than the MTU, so we will have to consider all these things in the future to see the real overhead percent. Other important thing we have to ensure is the Ip fragmentation. If there was IP fragmentation would mean that there would be more overhead since every packet has its own headers. Usually Ip fragmentation happens when all the transmitted data doesn't fit in one packet and it has to be divided in several segments that will be reassembled by the layer 3. It depends very much on the previous concepts of MSS and MTU. We will have to be careful when watching at wireshark traces because some of the overheads fields, like the FCS or th inter frame gap, are not showed by the wireshark frames. We should calculate the overhead from the post request and from the response as well

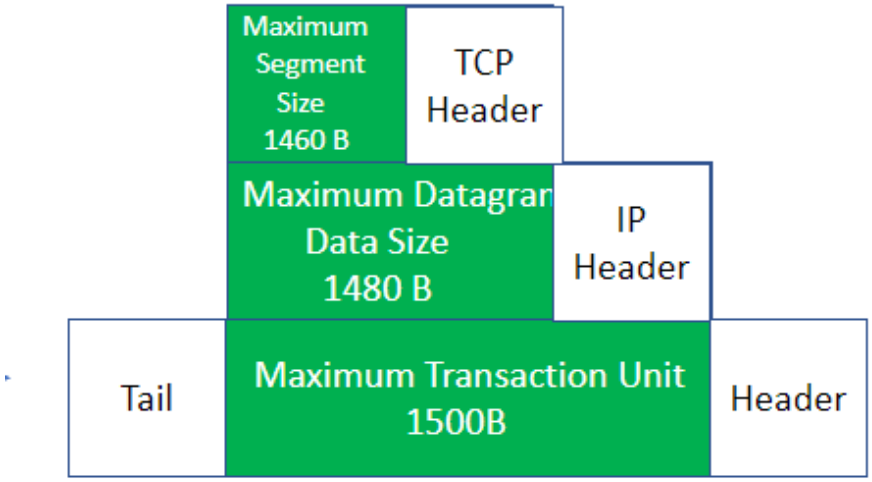


Figure 20: TCP structure segment schema, source: wireshark gui

because it may have overhead as well but it's needed a deeper research to get the values clear.

8.3.4 Securize the node server

By now the node server is open to anyone sending posts request through that endpoint. The optimal way of doing so would be that anyone who wanted to send data through it had to get through a previous authentication step through a user and a password, and our sever would respond with a token or similar method to get your board authenticated as a trusted client to our server.

8.3.5 Finish the ELK stack securization

As well as the node server securization, we must enable our kibana security features for a full equiped deployment. Until now our securized version of the ELK stack is halfway. Our stack is securized but still we have no connection to our node server but we have figured out what steps woulb be needed to do so. Luckily this doesn't look like it will take a lot of time but in the last weeks we had to write an article for the International Conference on distributed computer systems in Bologne that delayed all the final task scheduling and finally was hard to acomplish.

Appendices

Appendix A

Obstacles

During the development of the project, almost in every task of it, several obstacles have appeared. In the following lines we will make a brief resume of the most relevant ones.

A.0.1 Issues with the platformio IDE

Many times along the development of the thesis I have been needing to make a clean up of the libraries of the project. In the very beginning I didn't know what was the proper way to include library dependences, which derived in a very messed up project filesystem. Having to reorganize the whole project brought us to states where compiling the project was a tough job because deleting the dependences most of the times was not enough to rebuild it the proper way, I had to delete the library manually to let the platformio to fetch the right version again from the github account written down on the platformio configuration files. Some other times the platformio didn't fetch the new repository library by itself, so that an IDE restart was needed too. Until I understood this behaviour I lost a lot of time struggling with the environment and It was quite difficult to understand what was wrong with my code or the project.

A.0.2 Using platformio external libraries instead of Arduino Libraries

This was a very common issue at the beginning. In the first weeks I had the task of connecting via WiFi, the board to the Wifi, so inspired in a project from the subject IT Project, called CowLocalizer, I searched the name of the library on google and one platformio library appeared,then I installed it in my project, but when I tried to use it with the same code that worked on CowLocalizer, the connection never happened . I spend a lot of time trying different versions of

the code without getting any result. Luckily a comrade from de AC department who was working also with LoRaMesher on his TFG discovered where was our big mistake. There was no need to install any special library, since all the library we needed were already included in arduino framework, but having the same name was very likely to install it by mistake. Even more considering that for the first compilation you had to struggle installing many things, trying, until you got a functional version of the previous code. This first issue with the WiFi prevent us of making the same mistake again with the upcoming libraries that I had to use for other functionalities, for example for doing the http requests to transmit the data towards the server.

A.0.3 Conflicts with RadioLib and HTTPClient

When we were developing the module Network to handle the Wifi connections, and the Http requests towards the server and we tried to build it in the same project as the LoRaMesher, we realized that there was a conflict with the class HTTPClient which was in charge of doing the http requests. For some reason there was a class on RadioLib with the same name as the HTTPClient of the arduino framework. Reading documentation we got to know that this issue was already fixed on the latest version of RadioLib, we also saw that that module had no longer support since the functionality was the same as the already existing in Arduino framework and had no further advantatges. The problem was that LoRaMesher was using actually a modded version from RadioLib forked from a previous version of it . My first thought was trying to merge both branches, the modded version and the actual version of RadioLib where this issue was fixed. But it didn't give good results, the newer version of RadioLib had a lot of different things from the previous one and directly messed all up . Then after ponder a little bit more, we found a more simple solution. If there was a conflict with a certain class name, the only thing we had to do is just delete one of them and cross fingers that there will be no other class depending on it. So that's what we did, we delete the library HTTP from RadioLib, with the HTTPClient definition. There was only an include of this .h file in the main loramesher.h file. We delete it as well and no other problem seemed to appear so in fact nobody seemed to needed anymore. We had solved the issue.

A.0.4 The error called guru meditation error core 0 panic'ed (loadprohibited). exception was unhandled

This error appeared suddenly when everything was expected to work fine just seeing previous works of other students. The explanation of this was quite simple.

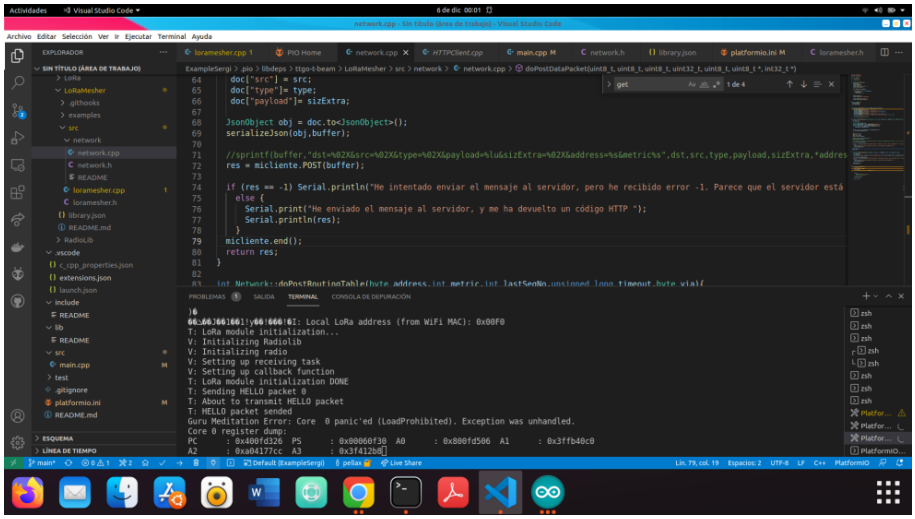


Figure 21: core 0 panic'ed (loadprohibited). exception was unhandled error screenshot,source: PlatformioIDE

This was caused by the board trying to execute a post or a get instruction without even being connected to the wifi. I assumed that it was already connected but for some reason I misunderstood the sequence of the events in the inherited code so by the the time that the GET instruction was executed, the WiFi connection still have not taken place which became in a core panic'ed error(fig 21).

A.0.5 Platformio doesn't flash more than one board at the same time

The problem is that Platformio not only doesn't flash more than one board at the same time, but it doesn't make you easy having the chance of which usb port you want to flash. The thing is that it took quite a lot of time to realize that because at the time of uploading the project to the board, Platformio in fact lets you to choose what serial you want to see, but I thought you were choosing what board to flash as well, but it did not happen like that, something just did not make sense. I realized it when some errors that I had already fixed seemed to appear again. Of course they did, because they were already working with the old version of the code without getting noticed. There are some ways you can flash multiple boards simultaneously, but you need to code a script in python but due to the deadlines I wasn't able to learn and test it properly by the time I found the obstacle. There are alternatives to get to flash many boards at the same time like executing the following python script described in list A.1

1

```
2 from platformio import util
3 import os
4
5 if name == 'main':
6     ports = util.get_serial_ports()
7     for port in ports:
8         os.system("pio run --target upload --upload-port " + port[
    "port"])
9         print("Successfully update port: " + port["port"])
```

Listing A.1: Python script to flash multiple boards,source :Joan Miquel Solé

A.0.6 Sensibility of the ELK stack to different versions between components and with the javascript client

During the development of the monitoring system I switched between different versions of elastic search and I must say that several issues may appear from doing so. There are some mechanism that remain unknown for me but I you switch to a branch with a lower elastic search version installed, when you try to turn on the systems, some pointers might be pointing the previous version, no matter if you are working with containers and you must be very careful if you are dealing with several versions on developing on the same machine.Afterwards I discovered that it was necessary to delete not only the containers of the previous docker deploys, but also the volumes executing the commands in list A.2

```
1 docker-compose down
2 docker rm -f \$(docker ps -a -q)
3 docker volume rm \$(docker volume ls -q)
```

Listing A.2: Previous recommended commands before a clean deploy from docker-compose file,source: [6]

Besides this, there's another important thing that to consider, the javascript client comparing to the ELK version, if they don't match the way the documentation says, several errors may appear, thread await errors, nodes that doesn't retrieve information or who knows. The things you should know about the javascript client [9]:
The client versioning follows the Elastic Stack versioning, this means that major, minor, and patch releases are done following a precise schedule that often does not coincide with the Node.js release times.
To avoid support insecure and unsupported versions of Node.js, the client will drop the support of EOL versions of Node.js between minor releases. Typically, as soon as a Node.js version goes into EOL, the client will continue to support that version for at least another minor release. If you are using the client with a version

Table 13: Node EOL versions,source: [9]

Node.js Version	Node.js EOL date	End of support
8.x	December 2019	7.11 (early 2021)
10.x	April 2021	7.12 (mid 2021)
12.x	April 2022	8.2 (early 2022)

Table 14: Compatibility matrix

ElasticSearch Version	Client Version
8.x	8.x
7.x	7.x
6.x	6.x
5.x	5.x

of Node.js that will be unsupported soon, you will see a warning in your logs (the client will start logging the warning with two minors in advance).
Unless you are always using a supported version of Node.js, we recommend defining the client dependency in your package.json with the ~instead of ^. In this way, you will lock the dependency on the minor release and not the major. (for example, ~7.10.0 instead of ^7.10.0). We can see all this better expressed in table 13
We should as well have in mind the compatibility matrix between the javascript client and our Elastic Stack, otherwise we will have several working issues appearing in many shapes.
Language clients are forward compatible 14; meaning that clients support communicating with greater or equal minor versions of ElasticSearch. ElasticSearch language clients are only backwards compatible with default distributions and without guarantees made [9] .

A.0.7 The amount of virtual memory

It's very common while working in dockerized environment, not having enough virtual memory space on the heap . Luckily this has a very quick fix executing the following instruction in list A.3 through the command line :

```
1 sysctl -w vm.max_map_count=262144
```

Listing A.3: Set virtual memory value instruction,source: [15]

A.0.8 The confusing placement of the documentation of Elastic Stack

When the new version of the Elastic Stack came out, the standard documentation of elastic search changed its version and all the older version documentation became more difficult to find. For some time I begun to think that it had disappeared. Luckily it was still there but somewhere hidden in elastic site but I lost some time exploring the newer version that I finally did not use. Afterwards I found the right documentation and after some research I could securize elastic side of the monitoring system. But that meant quite a lot of time until that.

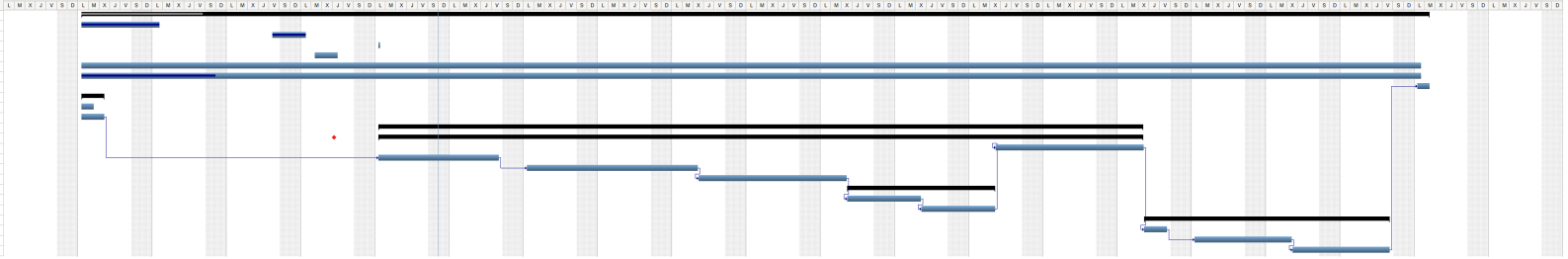
Appendix B

Initial Gantt Diagram

gantter

Project Name: **TFG**

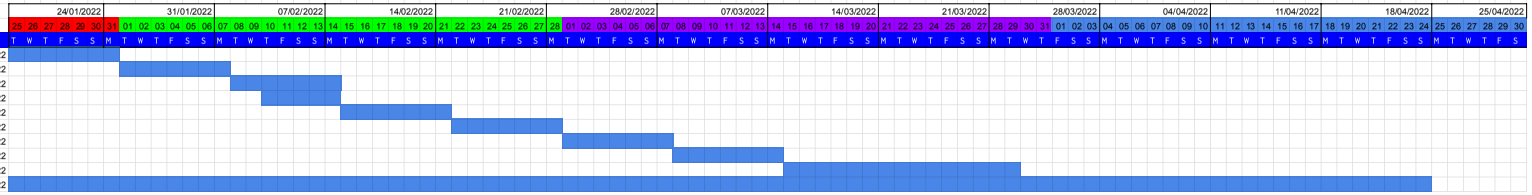
		Nombre	Duración	Inicio	Fin	Predecesoras	Recursos	Custom 1	Custom 2	Custom 3	Custom 4	Custom 5
1		**Project Management**	**183días?**	**09/13/2021**	**01/18/2022**							
2		Scope of the project	12días?	09/13/2021	09/29/2021							
3		Time planning	3días?	10/01/2021	10/04/2021							
4		Budget	1día?	10/11/2021	10/11/2021							
5		Sustainability report	5días?	10/05/2021	10/07/2021							
6		Meetings	182días?	09/13/2021	01/17/2022							
7		Documentation	182días?	09/13/2021	01/17/2022							
8		Presentation	3días?	01/17/2022	01/18/2022	24						
9		**Previous Work**	**5días?**	**09/13/2021**	**09/15/2021**							
10		State of Art	3días?	09/13/2021	09/14/2021							
11		Setting up the work environment	5días?	09/13/2021	09/15/2021							
12		**Developing**	**105días?**	**10/11/2021**	**12/22/2021**							
13		**Developing the board**	**105días?**	**10/11/2021**	**12/22/2021**							
14		D.1 Fin the protocol	2días?	12/06/2021	12/22/2021	20						
15		D.2 Connect the boards to the wifi	20días?	10/11/2021	10/22/2021	11						
16		D.3Transmit the logs	25días?	10/25/2021	11/18/2021	15						
17		Developing the server	20días?	11/19/2021	11/24/2021	16						
18		**Integrating parts**	**20días?**	**11/24/2021**	**12/08/2021**							
19		Containerizing the server	10días?	11/24/2021	12/01/2021	17						
20		Testing	10días?	12/01/2021	12/08/2021	19						
21		**Data treatment**	**35días?**	**12/22/2021**	**01/14/2022**							
22		Data collection	5días?	12/22/2021	12/24/2021	14						
23		Data preprocessing	15días?	12/27/2021	01/05/2022	22						
24		Graphics	15días?	01/05/2022	01/14/2022	23						



Appendix C

New Gantt Diagram

	Fecha de inicio	Fecha de fin
1Estudio nuevo código	25/01/2022	01/02/2022
2Implementación nueva tarea	01/02/2022	08/02/2022
3Conexión con servidor de Node en entorno dockerizado	08/02/2022	15/02/2022
3.1Test conexión con node	10/02/2022	15/02/2022
4Estudio ElasticSearch y Kibana	15/02/2022	22/02/2022
6Dockerización Kibana, ElasticSearch	22/02/2022	01/03/2022
7Integración node, mongo, elastic search	01/03/2022	08/03/2022
8 Experimentos Validación	08/03/2022	15/03/2022
9 Despliegue Ip Pública	15/03/2022	30/03/2022
10 Documentación adicional	25/01/2022	15/04/2022



	Start data	End data		
1Estudio nuevo código	0	7	25/01/2022	01/02/2022
2Implementación nueva tarea	7	7	01/02/2022	08/02/2022
3Conexión con servidor de Node en entorno dockerizado	14	7	08/02/2022	15/02/2022
3.1Test conexión con node	16	5	10/02/2022	15/02/2022
4Estudio ElasticSearch y Kibana	21	7	15/02/2022	22/02/2022
6Dockerización Kibana, ElasticSearch	28	7	22/02/2022	01/03/2022
7Integración node, mongo, elastic search	35	7	01/03/2022	08/03/2022
8 Experimentos Validación	42	7	08/03/2022	15/03/2022
9 Despliegue Ip Pública	49	15	15/03/2022	30/03/2022
10 Documentación adicional	0	90	25/01/2022	25/04/2022

Appendix D

Working with the new code

At the point where I was supposed to deliver my work I suffered some crucial delays and I had to renounce delivering my thesis at the end of January and leave till the end of april . On the other hand, there was another student called Joan Miquel Solé who coded an improved version of the LoRaMesher. This time it worked with no errors and all the test he did gave a good result. So I took advantage of his researchs and started working from that point. From now on I will explain the steps I took to get Joan Miquel code working on my old project on PlatformIO.

1. Copy paste from the platformio.ini main branch located at the example folder in the LoRaMesher project to the actual platformio.ini file.

2. Compile. Then the PLatformIO will download the LoRaMesher library with the right branch to the .pio folder and will change the name of the folder with the old LoRaMesher to avoid conflicts. Afterwards you can delete this library safely by hand because it won't be needed anymore.

3. Copy paste from the main.cpp located in the example folder from the LoRaMesher project to the actual main.cpp .

4. Compile

5. Upload to the TTGO-TBeam .Profit!

The next thing I did was creating my own branch from Joan Miquel version and I added my network.cpp library that was initially supposed to help extracting data from the old LoRaMesher towards our node server. A little modifications will be needed to get things working again with this new version.

Appendix E

Repos

Here are the main repos of the whole system

- LoRaMesher's side, concretely NuevaRamaAlejandro is my branch: LoRaMesher
- Node and elastic search server wihth dockerfiles here. It has several branches, halfway authenticated version, local test deployment, with three and one node, and version 8 test deployment.
- Main cpp used for flashing test boards with platformio here (Not included in the attached files because it was too big with all the library dependencies)

Bibliography

[1] BCN Wages list of the wages in the it world. https://www.glassdoor.es/Sueldos/barcelona-qa-sueldo-SRCH_IL.0,9_IM1015_KO10,12.htm?clickSource=searchBtn. Accessed: 2021-10-18.

[2] Dockerizing elk stack version 7.17. https://www.elastic.co/guide/en/elastic-stack-get-started/7.17/get-started-docker.html. Accessed: 2022-04-07.

[3] Elk stack. https://www.elastic.co/es/elasticsearch/. Accessed: 2022-04-14.

[4] Freertos http core. https://www.freertos.org/http/index.html. Accessed: 2022-04-14.

[5] Grafana features. https://grafana.com/grafana/. Accessed: 2022-03-17.

[6] How to do a clean restart of a docker instance. https://docs.tibco.com/pub/mash-local/4.1.0/doc/html/docker/GUID-BD850566-5B79-4915-987E-430FC38DAAE4.html. Accessed: 2022-04-15.

[7] Installing docker compose on ubuntu 20.04. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04-es. Accessed: 2022-04-15.

[8] Installing docker on ubuntu 20.04. https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-es. Accessed: 2022-04-15.

[9] Javascript elastic client. https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/installation.html?baymax=rec&rogue=rec-1&elektra=guide. Accessed: 2022-03-16.

[10] Kibana. https://www.elastic.co/es/kibana/. Accessed: 2022-04-14.

[11] LoRa Frequencies by country. https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country/. Accessed: 2022-04-2.

[12] Manually configure security. https://www.elastic.co/guide/en/elasticsearch/reference/current/manually-configure-security.html#manually-configure-security. Accessed: 2022-03-20.

[13] Maximum segment size definition. https://www.geeksforgeeks.org/how-to-calculate-maximum-segment-size-in-tcp/. Accessed: 2022-04-17.

[14] Maximum transmission unit. https://www.geeksforgeeks.org/what-is-mtumaximum-transmission-unit/. Accessed: 2022-04-17.

[15] Virtual memory in elastic. https://www.elastic.co/guide/en/elasticsearch/reference/7.17/vm-max-map-count.html. Accessed: 2022-04-15.

[16] What is Grafana. https://grafana.com/docs/grafana/latest/introduction/oss-details/. Accessed: 2022-03-17.

[17] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Mark Townsley. A study of lora: Long range amp; low power networks for the internet of things. *Sensors*, 16(9), 2016.

[18] Roger Pueyo Centelles, F. Freitag, R. Meseguer, and L. Navarro. A minimalistic distance-vector routing protocol for lora mesh networks. Technical report, Jun 2021.

[19] Antonio Cilfone, Luca Davoli, Laura Belli, and Gianluigi Ferrari. Wireless mesh networking: An iot-oriented perspective survey on relevant technologies. *Future Internet*, 11(4), 2019.

[20] Jeferson Rodrigues Cotrim and João Henrique Kleinschmidt. Lorawan mesh networks: A review and classification of multihop communication. *Sensors*, 20(15), 2020.

[21] Council of European Union. General data protection regulationgdpr, 2016. https://gdpr-info.eu/.

[22] Pengfei Hu and Wai Chen. Software-defined edge computing (sdec): Principles, open system architecture and challenges. In *2019 IEEE SmartWorld,*

Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pages 8–16, 2019.

[23] Dorian Pyle. *Data preparation for data mining*. morgan kaufmann, 1999.

[24] Francoise Sailhan, Liam Fallon, Karl Quinn, Paddy Farrell, Sandra Collins, Daryl Parker, Samir Ghamri-Doudane, and Yangcheng Huang. Wireless mesh network monitoring: Design, implementation and experiments. In *2007 IEEE Globecom Workshops*, pages 1–6. IEEE, 2007.

[25] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[26] Joan Miquel Solé. Improving the usability of a lora mesh library, 1 2022.

[27] Usama Fayyad Johannes Gehrke Jiawei Han Shinichi Morishita Gregory Piatetsky-Shapiro Wei Wang Soumen Chakrabarti, Martin Ester. DATA MINING CURRICULUM: A PROPOSAL. https://kdd.org/curriculum/view/introduction, 2021. [Online; accessed 26-September-2021].

[28] Er. Pooja Yadav, Er. Ankur Mittal, and Hemant Yadav. Iot: Challenges and issues in indian perspective. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–5, 2018.